

Study on the Software Collaboration Framework Based on Internetware

Hebiao Yang & Kai Chen

School of Computer Science and Telecommunication Engineering, Jiangsu University

Zhenjiang 212013, China

E-mail: tony_chen000@163.com

Abstract

With the popularization of the new computation environment, internet, traditional software form has gradually not adapted the development and application under the internet environment. So a new generation software form with many characteristics such as independence, collaboration, response, evolution and multi-object, internetware is proposed. Its establishment depends on the effective collaboration among various distributed and asynchronous autonomous software entities in the opening environment. In the collaboration process, one important problem is how to effectively plan and adjust the topology structure among software entities to realize users' demand in the dynamic environment. Aiming at this problem, several aspects including the organization and the assembling of the software entity and the evolvement of the topology are studied in this article, and concrete works include giving an internetware collaboration framework and flow model based on the Petri net and the mobile Agent technology, where the integrated model of internetware entities based on the Petri net reflects the ordered organization of the internetware entities in the problem space, and proving the support mechanism for the dynamically assembling among software entities based on the assembling model with the XML format, and using the influencing factors and the dynamic operation rules to describe several basic system evolvement activities.

Keywords: Internetware, Software system, Collaboration framework, Petri net, Mobil agent

At present, the computer application has completely entered into the age of internet, and the operation environment of the computer software has been turned from the static closing to the dynamic opening. But the characteristics of internet such as the "true" distribution without uniform control, the high self-government of nodes, the opening and dynamic characteristics of the node link, the indeterminism of entity behaviors, and the diversity of the network linking environment will largely influence the software system and profoundly change the traditional technologies and their application, operation and industry mode. At the same time, how to realize the sharing and integration of various resources and the development of the complex software system in the opening, dynamic and uncontrolled internet environment has been a challenging research task for the computer software technology.

The development history of the software method and the technology system shows that when the software base support platform and the corresponding users' demand change largely, the software method and the technology system will change with that, and provide drives for the occurrence of new methods and technology system, and promote the essential reform of the software engineering even the essential reform of the computer science criterion (Franco Zambonelli, 2003, P.329-342). To adapt this development tendency, WWW, the computational grid, large-scale resource sharing and integration with the sign of the network embedded system, multiple new computation modes such as the grid computation, the mobile computation, and the ubiquitous computing occurred subsequently (Foster I, 2004 & Cardelli L, 2000, P.3-37 & Fuggetta A, 1998, P.342-361 & Satyanarayanan M, 2001, P.10-17). But these technologies all have not solved the problems such as the independence and the collaboration of the software based on internet in the dynamic and opening environment. The software system based on internet can be regarded as the software alliance to form dynamically by a series of distributed independent computation resources and complete special tasks. Corresponding with that, the software system begins to present a new soft system form with multiple objects and continual reaction. As viewed from the technology, the software entities supported by the technologies such as internetware exist on various nodes of internet by the opening and independent mode, and any one software entity can link, communicate, collaborate and ally with other soft entities among networks by various collaborative modes in the opening environment, and perceive the dynamic change of the exterior network environment, and dynamically adjust and evolve it to make the system with more user satisfaction according to the function index, the performance index and the credit index with this change. A new software form, i.e. the internet ware has occurred in many articles (Yang, 2003, P.1104-1115 & Lv, 2006, P.1037-1080).

Concretely speaking, the internetware is a kind of abstract of the distribution software system in the opening, dynamic and uncontrolled network environment, and it can perceive the change of the exterior environment, and adapt this change and show corresponding context behaviors by the method of system structure evolvement (mainly by the adjustment of the system topology structure) (Lv, 2006, P.1037-1080).

As viewed from the software lifecycle, one important challenge in the analysis design stage and the online evolvement stage of the internetware system is how to design a proper mechanism to organize and assemble relative software entities and dynamically evolve the topology structure among software entities to satisfy users' demand according to the change of the environment.

The main works of this article include giving an internetware collaboration framework and flow model based on the Petri net and the mobile Agent technology, where the integrated model of internetware entities based on the Petri net reflects the ordered organization of the internetware entities in the problem space, and proving the support mechanism for the dynamically assembling among software entities based on the assembling model with the XML format, and using the influencing factors and the dynamic operation rules to describe several basic system evolvement activities. The main structure of this article can be described as follows. The first section is mainly to introduce the relative research results about the internetware, and the second section is to introduce the internet collaboration model and the system structure with the hierarchy style, and the third section is to mainly describe the basic principle and the core mechanism of the internetware collaboration framework including the internetware entity integration modeling based on the Petri net, the assembling model based on XML and the dynamic evolvement mechanism, and the fourth section is to summarize and discuss the further problems and directions.

1. Relative research results

The occurrence of the concept of internetware is not occasional, and with the quick development of the internet platform, people begin to study the distribution computer resources, the software form and the software development method, and put forward a series of new theories and methods. For the basic theories, British scholars put forward the concept of GUC (Global Ubiquitous Computer) in relative articles (Milner R, 2004), and GUC concluded the concept of Ubiquitous Computing (UC) and the concept of Global Computing. Shaw M et al also put forward the complexity and heterogeneity to screen the network, which could make us to interview and process various distributed information transparently among networks, and realized the concept of "the Network is Computer" (Steven D Gribble, 2001). For the new software form, Shaw et al proposed the concept of ORC (Open Resources Coalitions) from the resource integration and sharing, and the ORC means the temporary alliance formed dynamically by independently distributed resources facing special task, and these resources may be information, computing, communication, control or service (Raz O & Shaw M, 2000, P.159-170 & Shaw M, 2000 & Shaw M, 1999). The famous scholar of China, Academician Yang Fuqing first put forward the new software form, the internetware in his article (Yang, 2003, P.1104-1115). Professor Mei Hong of Beijing University also described this new software form in the article (Yang, 2008, P.818-828), and used the method of ABC (Architecture Based Computer Composition) to powerfully support the development of internetware. ABC is the software development method which takes the software reusing as the core idea, takes the software component as the basic entities, takes the software system structure as the center, and takes the software middleware as the running support, and it is the combination of the soft architecture and the component based software development. Scholar Lv Jian in Nanjing University proposed a kind of internetware model based on Agent (Lv, 2005, P.1233-1253), and established the technical approach of internetware model based on Agent, i.e. "the internetware model= opening collaboration model + environment driven model + intelligent credit model", which could establish the base for the further research. However, most of these works started from the macro angle, and few of them could support the development, deployment, and operation of the internetware system with the collaborative characteristic. The internetware collaboration frame in this article tries to make the modeling of the software entity service according to clients' demands by use the Petri net, and cut and integrate the functions of various software entities in the opening environment, and use the mobile Agent as the carriers to realize the collaborative mechanism, and dynamically evolve the internet entity integration model and the transition model by the change of the detective environment and the change of the demands, and finally realize the dynamic collaboration of the system in the opening environment.

2. System structure of the framework

2.1 Internetware collaboration model

The so-called "software collaboration" means the process to establish the communication association among software entities with certain principal characteristic, restrain the interaction of them and make them to work

harmoniously and achieve the appointed application aim (Lv, 2006, P.1037-1080). Traditional research about the software collaboration is to divide the application target into software entity behaviors. First, for the development, remark the entity according to the demands, compile the function modules, integrate various function modules and form a complete software, and when the demand changes, modify some modules to adapt the change of the demand. Second, for the structure, the software entities are closely coupling with the collaboration parts which are usually hidden in the software entities. This software collaboration model has good effect in the static and closed environment, but it doesn't fit for the opening, dynamic and uncontrolled environment. In the opening environment, the software design is not implemented in a uniform frame, and it can not confirm the infrastructure and the composing parts of the system before hand, and because of the dynamic characteristic of the environment and the decentralization of the management, different collaboration behaviors may be needed properly. Therefore, on the one hand, effective collaboration mechanism is needed to support, manage, and control the interactive behaviors of entities on internet, and on the other hand, sufficient flexibility should be offered to adapt the change of the environment and the demand. Based on the theoretical bases (Yang, 2003, P.1104-1115 & Lv, 2006, P.1037-1080 & Yang, 2008, P.818-828), the internetware collaboration model is proposed in this article. This model has two characteristics. First, it adopts the structure separating the software entity from the collaboration part to give prominence to the important of the collaboration parts, which makes the model to more smartly face the demand change and the opening environment. Second, it uses the mobile Agent as the collaboration medium, takes the internetware integration model based on the Petri net as the collaboration references, and offers good base for the dynamic evolvement of the infrastructure of the system. The internetware collaboration model is seen in Figure 1.

2.2 Frame structure

The frame system described in this article uses the idea of software reusing based on the software assembling as the references, and adopts the loose coupling structure separating the software entity part from the collaboration part, and the software entity part and the collaboration part are respectively developed by the supplier of the service and the integrator of the service independently (seen in Figure 2). This structure abandons traditional software structure which takes the close coupling structure, centralized development and the change model emphasizing programming as the characteristics. In the opening network environment, it can face the integrator's individual demands, give attention to the individual characteristics of the software entity, and comprehensively integrate various software services in the collaboration part, and continually statically adjust or dynamically evolve itself in the dynamic change of the exterior environment or the change of users' demand to enhance users' satisfaction.

(1) The application layer. It mainly concludes the description of user demand, and the function description of application software, and it adopts the mode such as function decomposing to organize the problem space and the user demand, and half-automatically generate the concept software infrastructure, which could smooth the gap between the demand and the design, and provide the reference for the organization and collaboration evolvement of the software infrastructure on the control layer.

(2) The control layer. It is the core of the software collaboration system, and the main content of this article. It is mainly composed by the collaboration engine module and the control part, and it is the intelligent part of the whole frame. The main work of the collaboration engine is to assemble the isolated software entities on the entity layer to the application software system with logic meanings by corresponding links, and the control part could supervise the user demand and the change of the environment in time, and feed back the structure to the collaboration engine. The design of this layer uses the knowledge such as the Web service combination, the dynamic evolvement of software system, and the mobile Agent for references, which could make the system to customize corresponding application software according to users' demand, and dynamically evolve the infrastructure with the change of the demand and the environment, and enhance users' satisfaction.

(3) The entity layer. It is the base of the control layer. And it concludes the link module based on mobile Agent and the software entities with distributed, independent, and heterogeneous characteristics in the opening environment.

3. Basic principle and core mechanism

3.1 Integrated modeling of internetware entities

Because of its stable mathematical base and network structure, the Petri net has excellent ability to describe the asynchronous concurrency and figure (Yuan, 2005 & David, 2005). The Petri net technology is adopted in this article to make the modeling for the entity integration. Following relative definitions are provided.

Definition 1: For the Petri net system, define a six-element set, $\Sigma = (P, T, F, K, W, M_0)$, and P is the set of all library places, T is the set of all transitions, $F \subseteq S \times T \cup T \times S$, and K, W, M_0 respectively are the capacity function, the weight function and the mark in Σ .

Definition 2: One software entity can be defined as one seven-element set in form, i.e. $SE = (Name, Desc, Loc, CS, State, Qos, SN)$.

Name: It is used to denote the name of a software entity, and it is the unique mark of the software entity.

Desc: It is used to express the service offered by the software entity.

Loc: It is used to record the position of the present software entity.

CS: It denotes an embedded software entity combination, and if $CS = (SE.Name)$, it denotes that SE is a basic service unit, or else, it denotes a combination of software entity.

State: State = (Active, Version, Available), and Active is a mark to denote whether a software entity runs, and Version denotes the present version number of the software entity, and Available denotes whether is the present software entity is available.

Qos: Qos = (T, C, Rep, Rel), and Qos represents the quality that one software offers service, and the factors influencing the service quality include the service response time T, the price C, the reputation Rep and the reliability Rel.

SN: $SN = (P, T, F, K, W, M_0)$, and it denotes the Petri net model expression of this software entity.

Definition 3: For the Petri net, the Petri net is the mapping from the entity system to the Petri net system, it is denoted by $\Sigma = (P, T, F, M_0)$, and it should fulfill following conditions. (1) $N = (P, T, F)$ is a net system. (2) P is the set of finite places, and it represents the set of all software entity input conditions and output conditions. (3) T is the set of finite transitions. (4) $F \subseteq (P \times T) \cup (T \times P)$ represents the arc set, and arc is used to describe the flow relation between place and transition. (5) $P \cup T \neq \emptyset$ $P \cap T = \emptyset$. (6) M_0 denotes the initial marking, and it decides the running state of the entity, and M is a mapping, $M : P \rightarrow \{0, 1, 2, \dots\}$, and $\forall p \in P: M(p) \geq 0$, i.e. $M(p) = 0$ or $M(p) \geq 1$.

Definition 4: IOPE matching. IOPE matching means to match the inputs, outputs, preconditions, and effects of the entity.

The similarity computation formula of IOPE is

$$\begin{aligned} sim_{IOPE}(Adv, Req) = \\ \mu_1 sim_{in}(Adv, Req) + \mu_2 sim_{out}(Adv, Req) + \\ \mu_3 sim_{pre}(Adv, Req) + \mu_4 sim_{effect}(Adv, Req), \\ \sum_{i=1}^4 \mu_i = 1 \quad -\Phi \end{aligned}$$

Where, sim_{in} , sim_{in} , sim_{in} and sim_{in} respectively are the similarity functions about the input set and the output set, and the computation method is the improved level matching algorithm. μ_i denotes the weight value. By above formula, the similarity computation of functions can be translated to the similarity computation of the service inputs (sim_{in}), outputs (sim_{in}), preconditions (sim_{in}) and effects (sim_{in}).

The meaning of IOPE matching is that by the similarity computations of the inputs, outputs, preconditions and effects of the software entity services, the software entity which can satisfy the demand of basic function can be acquired.

Qos matching is mainly to match the similarity of the response time, reliability, cost and reputation of the entity. The similarity function of Qos is

$$\begin{aligned} sim_{qos(i)}(Adv, Req) = \sqrt{sim_{min} sim_{avg} sim_{max}} \quad -\Theta \\ \text{Where, } sim_{min} = 1 - \frac{|\xi Adv.qos(i)_{min} - \xi Req.qos(i)_{min}|}{\xi Req.qos(i)_{avg}} \end{aligned}$$

$$sim_{avg} = 1 - \frac{|\xi Adv.qos(i)_{avg} - \xi Re q.qos(i)_{avg}|}{\xi Re q.qos(i)_{avg}}$$

$$sim_{max} = 1 - \frac{|\xi Adv.qos(i)_{max} - \xi Re q.qos(i)_{max}|}{\xi Re q.qos(i)_{max}}$$

$\xi (0 \leq \xi \leq 1, i=1,2,\dots,n)$ denotes the corresponding trueness of the i 'th dimensional service quality attribute, and it reflects user's dependence degree to the service quality.

$$sim_{qos}(Adv, Re q) = \sum_{i=1}^6 \xi Sim(Adv, Re q), \sum_{i=1}^6 \xi = 1, 0 \leq \xi \leq 1, i = 1, 2, \dots, 6.$$

By the definition 4 and definition 5, the software entity set with IOPE matching and Qos matching can be obtained, a Petri net modeling method based on target driver is proposed to make modeling for the collaboration process of software entity. The implementation approaches of the algorithm can be described as follows.

Algorithm 1: Establishing the internetware entity integration model.

Parameters: Software entity set A, target information set G, initial information set S, result set Result= Φ .

step1: Define $B = \bigcup_{A \in A} A_i(Q_i)$;

if $(B \supseteq G)$ goto step2,
else end;

step2: for each $G_i \in G$, do

{ $G = G - \{G_i\}$;

Search $A_j \in A$, do

if $((A_j(Q_j) \supseteq S_i) \&\& (Min(A_j(I_j) - S)) = true)$

// Judge whether A_j output includes the target information, and A_j input include most initial information.

{

Result = Result $\cup \{A_j\}$;

Temple = [$\bigcup_{A \in Result} A_i(I_i) - S \cup \bigcup_{A \in Result} Q_i(I_i)] \cup Temple$;

// Set the software entities which input information in Result is offered by the initial information set S into Temple

}

}

step3: if Temple = $\emptyset$, end;

else Result = Temple, goto step2;

step4: Modeling(Result);

// Establish the Petri net model for each element in the result set

Search $A_j \in A$, do

if $(A_j.input = null \parallel A_j.output = null)$ Delete A_j ;

// Delete isolated places without input information and output information

Assemble();

// Add initial places, target places and corresponding transitions to compose the Petri net system

The first step of this algorithm is to judge the execution of the algorithm according to the target information, and if the output information sets of all software entities can not completely include the target information set, the target information can not be implemented, and the algorithm ends. The second step and the third step are to find

out the set of the software entities which can realize the set of target information by a trace algorithm based on target. And the fourth step to the sixth step is to generate the Petri net system according to the set and the conversion rules of the software entities, and the fifth step could cut the function of the software step by deleting the isolated nodes.

3.2 Assembling model based on XML

The internetwork entity integration model obtained in the last section could provide reference for the dynamically assembling of the software entity in this section. Traditional component assembling technologies such as CORBA, DCOM, JavaRMI and so on are all based on classical C/S structure, and they are all deficient in the adaptability of the network environment including the independence, the flexibility and the subjectivity, and early confirmation of the entity link mechanism goes against the demands of the dynamic assembling of the entity in the internet environment. To realize the intelligent link of the software entities in the opening environment, the mobile Agent can be the implementer of the collaboration mechanism to realize the mobile Agent route information and the separated structure of the function entities, and generate the XML format describing the mobile Agent route planning information to instruct the behaviors of the mobile Agent. To better describe and analyze above technical route, relative definitions in this section are offered as follows.

Definition 4: Distribution transition. The transition with more than two input places is called as the distribution transition (seen in Figure 3).

Definition 5: Synchronization transition. The transition with more than two output places is called as the synchronization transition (seen in Figure 4).

Definition 6: The basic unit structure (BUS) is a Petri net system Σ , and when and only when

- (1) The net system Σ concludes one distribution transition and 1 synchronization transition;
- (2) The net system starts from the synchronization transition and ends by the distribution transition (seen in Figure 3).

Definition 7: For one BUS, if following conditions are satisfied

- (1) $\exists M \in R(M_0), M[t_1 > \wedge M[t_2 >$;
- (2) t_1 and t_2 are independent actions each other, i.e. $({}^{\circ}t_1 \cup t_1) \cap ({}^{\circ}t_2 \cup t_2) = \emptyset$,

t_1 and t_2 are called as two transitions with the concurrent relation, and are denoted by $t_1 \square t_2$.

Definition 8: For one BUS, if following conditions are satisfied

- (1) $\exists M \in R(M_0), M[t_1 > \wedge \neg M[t_2 >$;
- (2) $M[t_1 > \wedge M[t_2 >$,

t_1 and t_2 are called as two transitions with the sequence relation, and t_2 is the sequent action of t_1 , they are denoted by $t_1 \sqsubseteq t_2$.

By the modeling of the internetwork entity integration model in last section, the association among entities could be established according to the function demand. But this association is complex, and it goes against the reading by the computer, and it needs to be further abstracted. Based on the knowledge about Web Service, the association among software entities could be abstracted by the sequent assembling structure and the concurrent assembling structure, and these two assembling structures have sufficient representation ability in the collaboration mechanism based on mobile Agent to describe the transition process of the complex mobile Agent linkers. But for one complex internetwork system, the internetwork entity integration model based on Petri net will be very complex because it has numerous software entities and complex relations among entities, which will certainly increase the difficulty to deal with the entity integration Petri net. In this article, the method of processing the Petri net model layer by layer is adopted and the BUS is used to denote the complex Petri model, and one complex Petri net system can be denoted as many combinations of simple subnets with similar structure, which will finally enhance the efficiency and precision of the system. Based on relative article (Hyung Lee-Kwang, 1993), the algorithm of layer processing of the Petri net model has been obtained.

Algorithm 2: Internetwork entity integration model layer processing algorithm.

Step 1: Start from the initial place, find out the first BUS according to the definition 6, denote it as the Subnet 1, and connect the places among them and the transition sets by t_0 and t_1 ;

Step 2: Abstract the last level subnet Net_i as a common transition, and define the found BUS as the Subnet $SubNet_{i+1}$;

Step 3: Repeat Step 2, until all places and transitions could be concluded in various subnets;

Step 4: For each branch of each subnet, describe the sequent relation of each subnet according to the definition 7 and the definition 8 one by one.

By the algorithm 2, various subsystems of the layered Petri net can be described in form, and the forms are $SubNet_1 = ts \sqcap (t_1 \parallel t_2) \sqcap t_3$, $SubNet_2 = Net_1 \sqcap t_4 \sqcap t_7$ and $SubNet_3 = (Net_2 \parallel t_5 \parallel t_6) \sqcap t_e$. According to the formalized description of various subnets, two mobile semantics can be used to instruct the generation of the mobile Agent transition route, i.e. sequence and parallel. The sequence semantics denotes that the mobile Agent is transited to each entity one by one and exerts corresponding operation. The parallel semantics denotes that the mobile Agent derives n subagents which will be transferred to some entities in parallel, and exert corresponding operations. To directly use the formalized description of the Petri net system on the generation of the mobile Agent transition route, the formalized language needs to be converted to certain described language which can directly explain the execution. The XML language is adopted to design the mobile Agent transition model based on the definition of XML schema.

```
<schema targetNamespace="http://www.ujs.edu.cn/biz"
```

```
xmlns="http://www.w3.org/2001/XMLSchema">
```

```
<annotation>
```

```
<documentation xml:lang="zh">
```

```
schema for Internetwork Coordination Framework
```

```
</documentation>
```

```
</annotation>
```

```
<element name="entityName" type="string"/>
```

```
<element name="entityInput" type="string"/>
```

```
<element name="entityOutput" type="string"/>
```

```
<complexType name="entityQos">
```

```
<sequence>
```

```
<element name="reliability" type="elemType"/>
```

```
<element name="cost " type="comRef"/>
```

```
<element name="time " type="comRef"/>
```

```
</sequence>
```

```
</complexType>
```

```
<element name="subnetName" type="string"/>
```

```
<element name="parallel " type="string"/>
```

```
<element name="sequence " type="string"/>
```

```
</schema>
```

3.3 Dynamic evolution mechanism

In the system assembled by software entities, with the change of system maintenance and demand, the entities and linkers will change correspondingly, and this change will correspond with the behaviors of the software system evolvement, and accordingly embody the evolvement of the software system from the macro layer. But when the topology structure among entities and the corresponding linkers evolve, the total behavior of the system will be different with the present system. Therefore, it is the essential target of the entity system evolvement to ensure the coherence of the entity system. The reachable marking graph of the Petri net could analyze the state change and the transition happening sequence of one net system, so the relative characters of the net system could be obtained, and it is an important analysis tool of the Petri net. In the entity Petri net system, the reachable marking graph can easily define other entities influenced by certain entity change, and the dynamic performance rule of the Petri net can be used to simulate the harmonized process among software entities. The concepts of the Petri net dynamic performance rules and the concrete influencing factors are

described as follows.

Definition 7: Influencing factors. Influencing factors are used to denote the association degree of the node in one reachable marking graph, i.e. $\tau(M)$. The values of $\tau(M_i)$ is the amount of the reachable mark, i.e. the amount of different sequent marks of corresponding marks of various entities. $\tau(M)$ is bigger, it means the association degree of this peak is bigger, and the modifying influence of this peak is deeper. The influencing factors can be used to measure the spread effect of the entity evolvement to the entity system.

Definition 8: Dynamic performance rule. One entity Petri net system is a marking net $\Sigma = (P, T; F, M_0)$, and it has following dynamic performance rules. (1) For $t \in T$, if $\forall p \in P: p \in {}^\bullet t \rightarrow M(p) \geq 1$, so the operation t has the firing right in the mark M , i.e. $M[t >]$. (2) If $M[t >]$, so in the mark M , the operation t can fire, so the a new mark M' obtained from the mark M ($M[t > M']$) can be obtained, for $\forall p \in P$

$$M'(p) = \begin{cases} M(p) - 1, & \text{if } p \in {}^\bullet t - t^\bullet \\ M(p) + 1, & \text{if } p \in t^\bullet - {}^\bullet t \\ M(p), & \text{others} \end{cases}$$

The evolvement of the entity system can be regarded as following basic evolvement activities and their combinations, i.e. deleting entity, adding entity, replacing entity.

Deleting entity: Deleting entity C_i equals to deleting one transition t_i , relative inputs and output places in the entity Petri net, and in the reachable marking graph, if the relative place of t_i corresponds with the influencing factor of the mark $\tau(M_i) = 0$, i.e. this node is the dead node, and the corresponding entity and operations of this node mark can be directly deleted, and this operation can only change the structure of the entity system, i.e. it can only delete the function of the software, but will not influence other entities. On the contrary, if the relative place of t_i corresponds with the influencing factor of the mark $\tau(M_i) \neq 0$, i.e. this node is not a dead node, and the deleting of this node will induce the variance of the entity system structure, and it will impact the execution of the function of whole software, even produce the abnormal performance. Its influence is decided by the influencing factor $\tau(M_i)$ of the corresponding place t_i , and $\tau(M_i)$ is bigger, the influence is bigger.

Adding entity: Adding entity C_e equals to adding a corresponding transition t_i and relative places in the entity Petri net, and if it is possible, the relative operation transition of this entity is needed to add. This evolvement will generate a new entity Petri net system, $\Sigma(+p_e) = (P \cup \{p_e\}, T; F \cup F(p_e), M_0')$, and $F(p_e) \subseteq \{(p_e, t) | t \in T\} \cup \{(t, p_e) | t \in T\}$, and if new operation transition is needed to add, the new net system is $\Sigma(+p_e + t_e) = (P \cup \{p_e\}, T \cup \{t_e\}; F \cup F(p_e) \cup F(t_e), M_0')$. This operation will change the structure of the original entity system, and it is embodied in the adding of software function. The influencing factors of the mark and the achievable marking graph need to be rebuilt.

Replacing entity: Replacing entity C_i equals to replacing one transition T_i and corresponding places in the entity Petri net, and the spread effect of this operation is similar with deleting entity, i.e. if the influencing factor of corresponding mark of the relative place of t_i , $\tau(M_i) = 0$ so the corresponding entity and its operation of this node mark can be replaced directly, and if the interfaces of the replacing entity are matched better, this operation will not change the original system structure, but only induce the updating of the original system semantics and functions and the change of relative entity interaction. On the contrary, if the influencing factor of corresponding mark of the relative place of t_i , $\tau(M_i) \neq 0$, and the interfaces of updating entity are matched well, the spread effect of this operation is same with the former operation, but if this updating interfaces are not matched, its influence is decided by the influencing factor $\tau(M_i)$ of the corresponding place t_i , and $\tau(M_i)$ is bigger, the influence is bigger.

Above analysis about the entity evolvement have not considered the entity semantics and the spread effect of the entity static relationship, which is very necessary to analyze the evolvement of the entity system from the macro layer. The main content of the dynamic adjustment tactics in the system evolvements is to set up a threshold of the influencing factor by learning, and when the influencing factor of the evolvement activity exceeds this

threshold, it indicates that the change of the exterior environment will make the system to be changed in structure, and the system structure model needs to be rebuilt for ensuring the coherence of the system behaviors. When the influencing factor is in a controlled range, relative details of the system structure can be adjusted locally according to the influencing factors and the performance rules.

4. Conclusions

As a new software form, the internetware could more sufficiently exert the computation potential of internet, promote the enhancement of the software production efficiency and quality, deepen the application range of software, and possess wider research foreground. However, the research about this domain is still in the start stage at present, and relative theoretical researches are not matured, and some existing works only research and analyze from the macro angle, so the internetware still face many problems such as the collaboration organization of software entity, and the dynamic evolvement of software infrastructure. Aiming at these problems, the Petri net is adopted to describe the internetware entity integration model to provide the references for organizing and transferring relative software entities, and the mobile Agent is used to flexibly link various software entities including the route planning based on XML, and the influencing factors and the dynamic performance rules are used to describe several basic system evolvements.

This article mainly emphasizes the theoretical research of the internetware collaboration mechanism. And the works mainly include realizing the algorithm in this article, establishing corresponding analysis and validation ante-type tool, and empirically studying the software systems with some internetware characteristics occurring at present. But there are many problems which have not been solved. For example, for the method layer, the more complete methodology needs to be provided for supporting the internetware collaboration mechanism, and for the platform layer, more abundant emergency processing ability and more perfect dynamic adjustment strategies are hoped.

References

- Cardelli L. (2000). *Mobility and security*. In: BauerF L, Steinbrueggen R, eds, *Foundations of Secure Computation*. Amsterdam: IOS Press. P.3-37.
- David, René, Alla. (2005). *Discrete, Continuous, and Hybrid Petri Nets*. Springer Verlag.
- Foster I, Kesselam C. (2004). *The Grid: Blueprint for a New Computing Infrastructure (2nd ed)*. Amsterdam. London: Morgan Kaufmann.
- Franco Zambonelli. (2003). Towards a paradigm change in computer science and software engineering. *The Knowledge Engineering Review*. No. 18(4). P.329-342.
- Fuggetta A, Picco P G & Vigna G. (1998). Understanding code mobility. *IEEE Trans Software Eng*. No. 24(5). P.342-361.
- Hyung Lee-Kwang & Joel Favrel. (1993). Representation of Nonstructured Concurrency by Petri Net Languages. *IEEE Transactions on Systems, man, and Cybernetics*. Vol 23, No.3.
- Lv, Jian, Ma, Xiaoxing & Tao, Xianping. (2006). Research and Development of Internetware. *Science in China (Series E)*. No. 36(10). P.1037-1080.
- Lv, Jian & Tao, Xianping et al. (2005). Study on the Internetware Model Based on Agent. *Science in China (Series E)*. No. 35(12). P.1233-1253.
- Mei, Hong & Huang, Gang et al. (2006). An Internetware Development Method Taking the Software System Structure as the Center. *Science in China (Series E)*. No. 36(10). P.1100-1126.
- Milner R. (2004). *Theories for the global ubiquitous computer*. In: *Foundations of Software Science and Computation Structures*, LNCS 2987. Berlin:Springer-Verlag.
- Raz O & Shaw M. (2000). *An approach to preserving sufficient correctness in open resource coalitions*. In: *Proceedings of the 10th International Workshop on Software Specification and Design*. Washington DC: IEEE Computer Society. P.159-170.
- Satyanarayanan M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Commun*. No. 6(8). P.10-17.
- Shaw M. (1999). *Architectural requirements for computing with coalitions of resources*. In: *Proceedings of the First International Federation for Information Processing Conference on Software Architecture*. Cambridge: ACM Press.

Shaw M. (2000). *Sufficient correctness and homeostasis in open resource coalitions: How much can you trust your software system*. In: Proceedings of the Fourth International Software Architecture Workshop.

Steven D Gribble, Andrew Whitaker & Marianne Shaw. (2001). Lightweight Virtual Machines for Distributed and Networked Systems, Talks in HP Labs and Sprint Labs. 2001-07-07.

Yang, Fuqing, Lv, Jian & Mei, Hong. (2008). Internetwork Technology System: An Approach Taking the System Structure as the Center. *Science in China (Series E)*. No. 38(6). P.818-828.

Yang, Fuqing, Mei, Hong & Lv, Jian. (2003). Some Discussion on the Development of Software Technology. *Acta Electronica Sinica*. No. 26(9). P.1104-1115.

Yuan, Chongyi. (2005). *Principle and Application of Petri Net*. Beijing: Electric Industry Press.

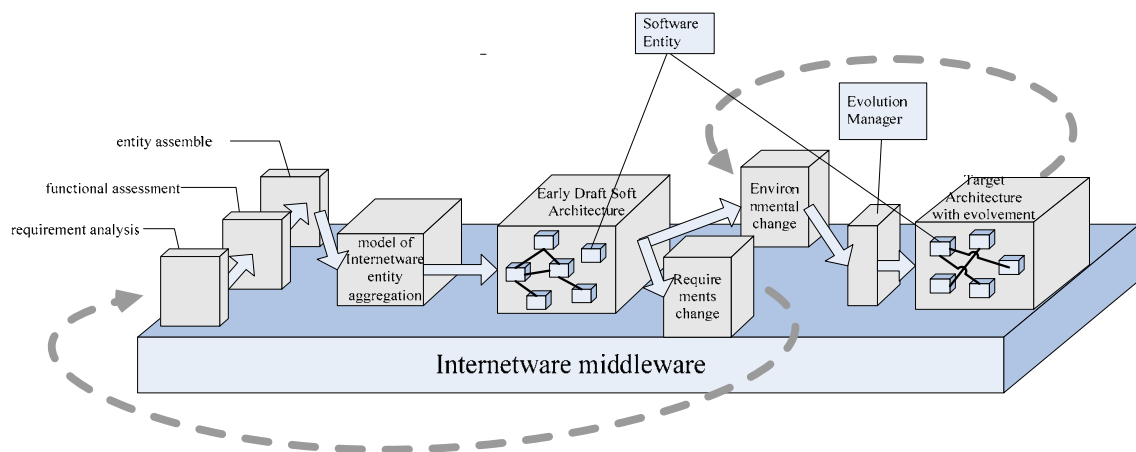


Figure 1. Internetwork Collaboration Model

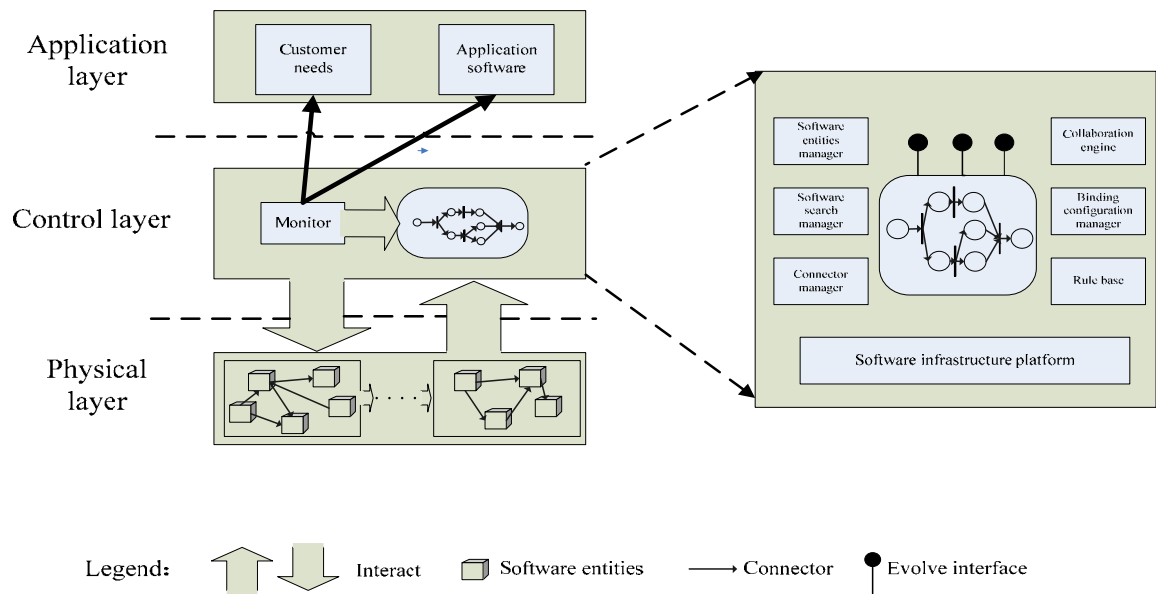


Figure 2. System Structure of the Software Collaboration Framework Based on Internetwork

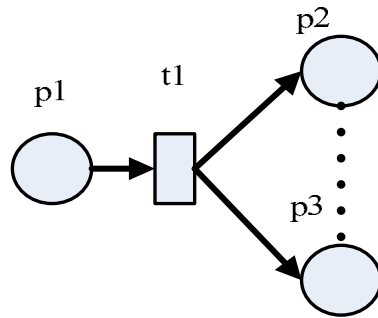


Figure 3. Distribution Transition

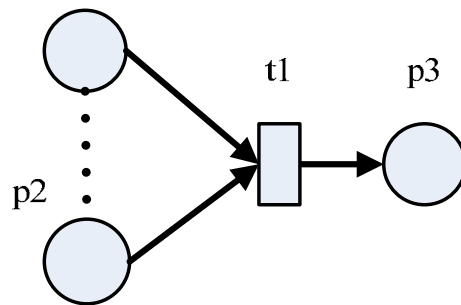


Figure 4. Synchronization Transition

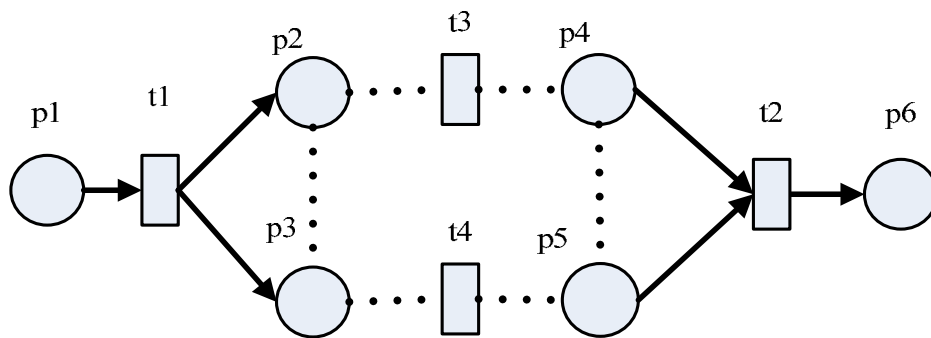


Figure 5. Base Unit Structure BUS