Dynamic Capability Delegation Model for MAS, Architecture and Protocols in CSCW Environment

Zhiyong Zhang[1], Jiexin Pu[1], Shaozhong Zhang[2]
[1] *Electron.Inf.Eng.Coll., Henan University of Science & Technology, Luoyang, P.R.China*
zhangzy@mail.haust.edu.cn; pjx@mail.haust.edu.cn
[2] *Electron.Eng.Dept., Zhejiang Wanli College, Ningbo, P.R. China*
dlut_z88@163.com

Abstract

The purpose of delegation is improving resource share and realizing collaboration in CSCW environment. Nowadays agent-based collaboration in Multi-Agent System lacks of delegation model including formalization and relative mechanisms, especially for variable dynamic collaboration scene. Based on a new proposed concept Agent Collaboration Scene (abbr. ACS), this paper specifies a dynamic agent-based capability delegation model supporting temporal characters and constraint rules, called by Capability Delegation Model for Multi-Agent System (abbr. CDM for MAS), as well as representing hybrid architecture and key protocol functions related to delegation. The model solves the issue of agent's capability oneness and inflexibility, and realizes dynamic capabilities delegation between agents in an actual application Agent-Based Text Information Retrieval System, enhancing agent cooperation efficiency and security.

Keywords: *CSCW, Multi-Agent System, Agent Collaboration Scene, Capability Delegation, CDM for MAS*

1. Introduction [1]

Nowadays the theory and technology of Multi-Agent System (abbr. MAS) are hot topics in Distributed Artificial Intelligent research. Multi-agent collaboration research focuses on the architecture and how to realize dynamic schedule, avoid collision, harmonize each other and cooperative work, and assure higher efficiency, security and stability of MAS. However agent's capability is single or stable in traditional MAS, so it is not fit for dynamic, complexity, and uncertainty of MAS [1]. Accordingly, an incompact federal framework was presented in Literature [2], achieving distributed problem's computing; Andrea Omicmi introduced a role-based multi-agent cooperation model, architecture and related functions, but it lacked of model formalism [3]; Lai also created a role-based multi-agent workflow model for WFMS, and represented it using BNF extensions [4]. Because these above mentioned models do not deal with collaboration management, especially for delegation technology and related mechanisms, it is not fine-grained.

Summarily speaking, as the essential component and key technology of fulfilling multi-agent collaboration in CSCW environment, now there is little research and application on delegation. But some related research has progressed, for example, some issues and methods of role-based collaboration were represented by Haibin Zhu, such as role assignment and migration, role coordination, role collision and so on [5]. Anand Tripathi realized role model and domain policy that meet dynamic security, but a formal security model was not specified [6]. A role-based access control model for CSCW and a role-based multi-agent management model were formally defined, and it focused on basic component and general authorization rules, not dealing with delegation mechanism [7, 8, 9]. According to dynamic, complex, variable environment of MAS, this paper introduces role-based policy and ACS, and specifies a dynamic delegation model, architecture and key protocols of agent's capability delegation, realizing flexible capability delegation in an application.

2. Agent and Delegation Fundamental Theories

2.1. Agent Characteristics and Multi-Agent Collaboration

Intelligent agent has some main characteristics of autonomy, social ability, reactivity and benevolence. These embody that agent could apperceive exoteric environment and react, further affect on real world. The capability of collaborative computing in MAS exceeds

[1] This paper is sponsored by National Nature Science Foundation of China (Grant No.60475021) and Henan Province Natural Science Fundamental Research Fund (Grant No. 200410464004).

single agent, so it is a main reason of designing and realizing MAS. Compared with single agent, MAS has some features as follows: every agent only possesses partial system information and computing capability, so its viewpoint is limited. Here agent can not control a whole system, data are dispersive or distributed, and computing procedure is asynchronous, concurrent or parallel. But the structure of MAS could be pure centralized, whole distributed or hybrid form [1]. Because MAS is complicated and dynamic, its collaboration management is crucial to multi-agent harmony, negotiation, cooperative work in agent society.

2.2. Delegation Model and Main Features

The basic idea of delegation is that active entity (user, process, agent, et. al.,) in application could grant some own permissions or roles to others, which can carry out some privileges and functions on behave of the former. The concepts related to delegation have delegator, delegated role or permission and delegatee.

Delegation has some important features as follows:

1. Delegation Granularity: The unit of Delegation has three kinds as follow: Permission-based thin granularity [9], role-based medium granularity [10, 11], permission and role -based fat granularity proposed by Zhang [12, 13]. Thin granularity means that user could delegate the partial permissions of a role to delegtee, not just whole role. So granularity is depressed, and it meets the principle of least privilege, but brings about some non-integrity role led to authorization complexity .For medium granularity, delegtor only could delegate role as a whole, thus delegtee would acquires entire permissions of role. Apparently, it sacrifices the principle of least privilege at some degrees to eliminating non-integrity role. Fat granularity allow user to delegate own permission or role discretionarily, which is flexible compared with above two granularities besides of complex realization. With regard to delegation unit, it should be chosen according to applied system.

2. Delegation Step: It is subdivided into single-step delegation and multi-steps delegation. The former is that delegatee could not delegate role or permission to others further; the latter allows delegatee to grant further, but in the condition revoke is more complicated.

3. Delegation Revoking: The contrary operation of delegation is revoking which means that delegated roles or permissions are called off. Revoking mainly includes the following features, such as cascading revoke, non-cascading revoke, grant-independent revoke, grant-dependent revoke, system automatic revoke and user revoke.

3. CDM for MAS and Formal Definitions

3.1. Basic Ideas

CDM for MAS is a reference model with delegation feature for multi-agent cooperation, it introduces two new concepts of agent's role and dynamic ACS on the basis of role-based policy. The policy simplifies multi-agent society management, agent not only could active new role, further acquire other cooperative abilities, but also could delegate its capability or role to another agent, improving information sharing and cooperative work. Realizing collaboration constraint in ACS also ensures MAS security. Here delegated capabilities or roles compose a set called as Delegated Capability/Role Set (abbr. DCRS).The model contains some basic components, such as agent, role, DCRS, capability, task, activity, society and constraint, as Figure1.

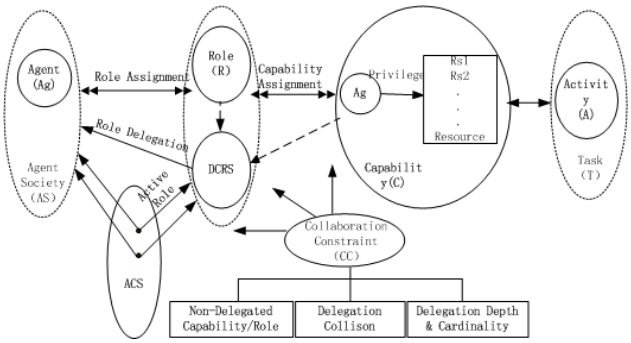

Figure 1. CDM for MAS

3.2. Main Components Formalization

The following is a list of CDM for MAS components: Ag, R, AR, DCRS, S, C, Rs, P, CC, AS, T, A are respectively defined as sets of agent, role, active role, delegated capability/role set, states, capability, resource, privilege, collaboration constraint, agent society, task, activity.

Definition 3.2.1 (Agent): Agent is an entity that has knowledge, faith, intention and expectation. It could apperceive the environment change independently, and make action, further effect environment. So it is a subject with autonomic and social properties. Agent is presented as a seven tuple that consists of AID, sensor, processor, communicator, aim, effector, collaboration scene.

$\forall$ ag (ag $\in$ Ag) ag = { aid, sensor, processor, communication, aim ,effector, acs}

Definition 3.2.2(Capability): Capability is a privilege, permission or condition accessing resource for a task. It could be not only actual read, write a file, but also abstract ability like delegation, revocation, etc. Every capability must and only accomplish an activity. Here

defines a function "doactivity()" to denote a one–to-one relation between capability and activity.

$C \subseteq RO \times P$

doactivity (c) : $C \rightarrow A$

Definition 3.2.3(Role): Role is an abstract entity who has definite responsibilities and capabilities in an organization or society, agent who acts as a role becomes an instance of the role. Besides role is looked upon as a set of agents, it is also a finite set of capabilities. The relation between role and agent is a many-to-many relation.

$\forall r (r \in R)\ r = \{ag_1, ag_2, \ldots ag_n \mid ag_i \in Ag\}$

$\forall r (r \in R)\ r = \{c_1, c_2 \ldots c_m \mid c_i \in C\}$

Definition 3.2.4(Society): Society is an organization where some agents or roles cooperate for a goal or task.

$\forall s (s \in S)\ s = \{ag_1, ag_2, \ldots ag_n \mid ag_i \in Ag\}$

$\forall s (s \in S)\ s = \{r_1, r_2, \ldots r_n \mid r_i \in R\}$

Definition 3.2.5(Agent Collaboration Scene): ACS is a dynamic agent cooperative environment that is collaboration context. It includes cooperation time, space, partner, shared resource and attributes that are defined in following Section 3.5.

$\forall$ acs (acs$\in$ ACS) acs = {time, space, parter, resource, attributes}

Definition 3.2.6(Role Assignment): According to cooperative task, agent is assigned to proper roles, and has some related capabilities. RA is a relation of many to many between agent and role.

$RA \subseteq Ag \times R$

R(ag)={r | role of assigning a agent ag, ag$\in$ Ag}

Definition 3.2.7(Active Role): It is a role that is activated by Agent in a given ACS, AR(acs) denotes active role(s) in ACS. Agent work through a mapping function "agentwork()" between Ag and AR.

$\forall$ acs (acs$\in$ ACS) (ar(acs)$\in$R(ag))

agentwork (ag,ar):$Ag \rightarrow AR$

Definition 3.2.8(Task & Activity): Agent roles' collaboration goal is called as task, the every step of task is called activity. Activity has dynamic and atomic characteristic, and is basic unit of task.

$\forall t (t \in T) t = \{a_1, a_2 \ldots a_n \mid a_i \in A\}$

Property 3.2.1(Agent Society-Task): Every agent society only has a task. Predication "t(s)" denotes the task of society s.

$\forall s (s \in S)\ \exists t_1, t_2 (t_1, t_2 \in T)\ t_1(s) \wedge t_2(s) \rightarrow t_1 = t_2$

Definition 3.2.9(Capability Assignment): CA denote assigning appropriate capabilities to role based on responsibility or task of role, and it is many-to-many between role and capability, written as a two tuple CA(r, c).

$CA \subseteq R \times C$

CA(r, c) = {c_i | capability assigned to a role r, r$\in$ R, c$\in$ C}

Definition 3.2.10(Delegated Capabilities/Role Set): DCRS is a set of delegated capabilities or (and) roles. Because role is treated as capabilities set, the granularity of DCRS is capability.

DCRS = $\{c_1, c_2, \ldots c_m, r_1, r_2, \ldots r_n \mid c \in C, r \in R\}$, namely, $DCRS \subseteq R \cup C$

Definition 3.2.11(Capability Delegation): Delegation relation is a six tuple (Ag_1, Ag_2, DCRS, DTL, CC, ACS) ,where Ag_1 is delegator, Ag_2 is delegtee, DTL is the limitation set of delegation time, and CC is the conditions set of capability delegation constraints. The relation is that agent Ag_1 could delegate DCRS to another agent Ag_2, thereby Ag_2 acquires whole explicit and implicit capabilities of DCRS, where delegation must meet prerequisite conditions of CC in an ACS.

Property 3.2.2(Cascading Delegation Revoke): When original user revokes delegation or ST exceeds DTL, the whole roles and capabilities of DCRS which is multi-steply delegated will be revoked cascadely.

Property 3.2.3(Grant-Independent Revoke): Every delegator in the delegation path could revoke delegation, not only original delegtor has right to revoke.

Property 3.2.4(System Imperative Revoke): When system clock exceeds to DTL of DCRS or entities' attributes changes, MAS system automatically revokes delegation including all explicit and implicit capabilities.

3.3. Delegation Temporal Properties

Definition 3.3.1(States Set): The states set of DCRS S={init, invoke, sleep, expire}, init is beginning-state, invoke is active-state, sleep is sleepy-state, expire is exiting-state.

Definition 3.3.2(Delegation Time Limit):DCRS has the property of time limitation, DTL={x | $x=[\tau_{bi}, \tau_{ei}](i=1,2 \ldots n)$}, where τ_{bi} is begin-time , and τ_{ei} is end-time.

Definition 3.3.3(State Transitions):ST is system time, $\forall i(i \in N)\ ST \notin [\tau_{bi}, \tau_{ei}] \wedge ST < \tau_{b1} \rightarrow S=init$; $\exists i (i \in N) ST \in [\tau_{bi}, \tau_{ei}] \rightarrow S=invoke$; $\forall i(i \in N) ST \notin [\tau_{bi}, \tau_{ei}] \wedge (ST > \tau_{b1}) \wedge (ST < \tau_{en}) \rightarrow S=sleep$; $\forall i(i \in N) ST \notin [\tau_{bi}, \tau_{ei}] \wedge ST > \tau_{en} \rightarrow S=expire$.

Property 3.3.1(Activity Temporal Order): The relation of activities is synchronization or concurrency. Between every two activities exits steady time order that meet partial order relation, denoted by "□".

$\forall t, a_i, a_j (t \in T, a_i, a_j \in A)(a_i \in t \wedge a_j \in t \rightarrow a_i \square a_j)$

$\forall t, a_i, a_j, a_k (t \in T, a_i, a_j, a_k \in A)(a_i \square a_j \wedge a_j \square a_k \rightarrow a_i \square a_k)$

Property 3.3.2(DLC Run-Order): Delegated Capabilities (abbr. DLC) are executed in series or

concurrently, and their relation is partial order, denoted by "$\geqq_{DC}$".

$\forall$ t,dlc_i,dlc_j,dlc_k(t $\in$ T,dlc_i,dlc_j,dlc_k $\in$ t)($dlc_i \geqq_{DC} dlc_j \wedge dlc_j \geqq_{DC} dlc_k \rightarrow dlc_i \geqq_{DC} dlc_k$)

Property 3.3.3(Activity-DC Temporal Order Consistency): Activity and DC is consistent in time order owing to their one to one mapping relation. Predication "own (a_i,dlc_j)" denotes activity a_i owns delegated capability dlc_j.

$\forall$ t,a_i,a_j,dlc_i,dlc_j(a_i,$a_j$$\in$ t)(own(a_i,dlc_i) $\wedge$ own(a_j,dlc_j) $\wedge$ $a_i \square a_j \leftrightarrow dlc_i \geqq_{DC} dlc_j$

3.4. Delegation Constraint Rules

Definition 3.4.1(Non-Delegated Capability/ Role): NDCR is a set of whole capability and roles that are not delegated to others.

NDCR={ c_1,c_2,…c_i, r_1,r_2,…r_k}

Definition 3.4.2(Delegation Collision Capability): Two capabilities c_i and c_j are delegation collision capabilities, if they are not delegated to a agent ag_k at the same time, we denote it by collc(c_i, c_j, ag_k).

Definition 3.4.3(Delegation Collision Role): Two roles r_i and r_j are delegation collision roles, if they are not delegated to a agent ag_k at the same time, we denote it by collr(r_i, r_j, ag_k).

Constraint Rule 3.4.1(Non-Collision Delegation Constraint): The elements of NDCR can not be delegated.

$\forall$ x(x$\in$ NDCR) $\rightarrow$ (x$\notin$ DCRS)

Constraint Rule 3.4.2(DCRS Non-Collision Constraint): Every two capabilities or roles do not exist delegation collision in DCRS set.

$\forall$ c_i,c_j ($c_i$$\in$ DCRS, $c_j$$\in$ DCRS, $ag_k$$\in$ Ag) collc(c_i, c_j, ag_k)=F

$\forall$ r_i,r_j ($r_i$$\in$ DCRS, $r_j$$\in$ DCRS, $ag_k$$\in$ Ag) collr(r_i, r_j, ag_k)=F

Definition 3.4.4(Delegation Depth and Cardinality): Delegation Depth d is a natural number about cascading delegation degree of capability c_i. It is called as single-step delegation if d=1, and called as multi-step delegation if d>1.Delegation cardinality is also a natural number about delegated agents' number of capability c_i.

Constraint Rule 3.4.3(Multi-Steps Delegation Constraint): Delegation depth and cardinality of DCRS lie on minimum delegation depth and cardinality of all roles and capability in DCRS.

$\forall$ c_i ($c_i$$\in$ DCRS) $d_{DCRS}<d_{ci} \wedge d_{DCRS}<d_{ri}$

$\forall$ r_i ($r_i$$\in$ DCRS) $n_{DCRS}<_{ri} \wedge n_{DCRS}<n_{ri}$

Property 3.4.1(Multi-Steps Delegation): DCRS multi-steps delegation is a partial order relation with reflexive, antisymmetric, transitive properties.

3.5. Delegation Dynamic Features

Definition 3.5.1(Attributes): Attribute represents characters of system entities including delegator Agent, delegatee and resource, respectively defined as Dlgtor_Attr, Dlgtee_Attr and Reso_Attr. Usual main attributes of Dlgtor_Attr and Dlgtee_Attr have identity, role, security level, capability scope; Reso_Attr includes access permission, share property and access cardinality. Attributes' change in ACS directly influences delegation constraint rules, delegation verification and access control.

Definition 3.5.2(Delegation Verification): After delegator receives the delegation request of delegate, system should make delegation verification according to relative attributes and ACS.DV usually acts on client. Here defines two operations related to delegation rules: rule consistency operation ▲, contrary operation ▼, and presents the definition using BNF extension.

```
<DeleVerification>::="Dlg_Veri"
  <input>::=<Dlgtor><Capability><ACS Contexts>
  <output>::=<Allow>|<Reject>
   {
     <Client>: if Dlgtor_Attr && Dlgtee_Attr && Reso_Attr &&
     Capability▲ Constraint Rules && ACS then
              {send_result(Allow);
               Delegation;}
            else
              send_result(Reject)
            end; }
End;
```

Definition 3.5.3(Capability Access Decision): When delegatee begin to accesses resource, system must make decision on capability of resource access based on Delegation Certificate and ACS, and send decision result to delegatee.

```
<AccessDecision>::="Access_Deci"
<input>::=<Dlgtee><Resource><Dlgtee_Attr><ACSContexts><Cap
ability>
  <output>::=<Allow>|<Reject>
   {
     <Server>: if Dlgtor_Attr && Dlgtee_Attr && Reso_Attr &&
     Capability ▲ Authorization && ACS then
              {send_result(Allow);
               Access Resource;}
            else
              send_result(Reject)
            end;  }
End;
```

Definition 3.5.4 (Delegation Validity Test): In the procedure of delegatee access to resource through DLC, system and delegator should test the validity of DLC periodically because of some change of attributes and ACS. If variable attributes do not fit for constraint rules or ACS, system or delegator will automatically revoke delegated capability.

```
<AccessTest>::="Access_Test"
```

```
<input>::=<Dlgtee><resource><TimePeriod><Cap-
ability>
  <output>::=<Record>|<Revoke>
   { for(;;TimePeriod)
       <Server>/<Client>: if Dlgtor_Attr && Dlgtee_Attr &&
     Reso_Attr && Capability ▲Constraint Rules && ACS then
              Record this test;
           else
              Revoke (Capability);
              Record this test;
           end; }
End;
```

4. Hybrid Architecture and Key Protocols

The architecture of CDM for MAS adopts Client-Delegation Reference Monitor(C-DRM) and Server-Delegation Reference Monitor(S-DRM) hybrid pattern as Figure 2.Client runs C-DRM that is Trusted Computing Base of delegation verification and test, it also stored a copy of capability assignment. C-DRM verifies delegation according to relative rules, then subscribes Delegation Certificate (abbr. DC) and sents it to Delegation Database as audited information, so fulfills delegation process. At last, C-DRM replies to delegate. Resource sever adopts S-DRM that is the TCB of access control and test. According to entities' attributes, S-DRM and C-DRM test DLC validity periodically. If delegation has invalidation, server pause the capability promptly and C-DRM revokes delegation. Delegatee could access to shared resource acting on delegated capability through submitting DC and ACS context. As complementary function, audit is in charge of track of agent's access and DC validity.

Because resource storing patterns could be centralized, federal or distributed frame, S-DRM is also the same configuration. Here only gives common centralized shared resource architecture.

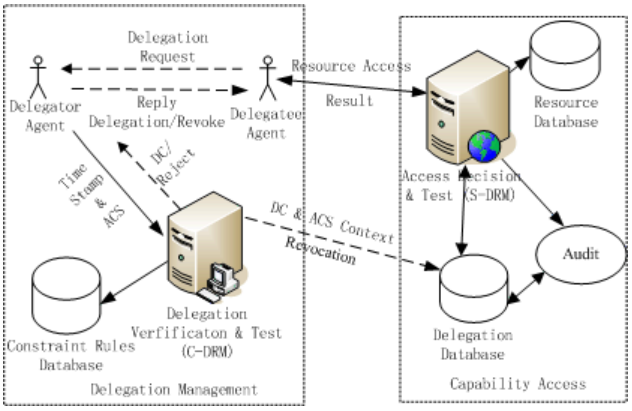

Figure 2. Hybrid architecture of centralized resource

The key protocols and relative semantics of this model are illustrated in Table1, for example, delegation request function DeleRequest; delegation verification function DeleVerify; delegation answer function DeleReply; delegation function Delegation; revoke function Revoke; these functions belong to C-DRM; Test function DeleValiTest acts on C-DRM and S-DRM. Besides, access decision function AccessDecision belongs to S-DRM.

Table 1. Key protocols functions

Main Entities	Key Protocols	Relative Semantics
Dlgtee, C-DRM	DeleRequest(dlgtor,dlg-ee, resource, capability)	Delegatee requests delegated capability of delegator.
Dlgtor, C-DRM	DeleVerify(dlgtee,dlgt-or,capability, ACS)	Delegation Verifi-cation on C-DRM integrating ACS.
Dlgtor, C-DRM	DeleReply(dlgtor,dlgtee,re sult)	Sending result to delegatee.
Dlgtor, Dlgtee, C-DRM	Delegation(dlgtor,dlgtee,D C, timestamp)	Dlgtor and dlgtee subscribe DC on C-DRM that sends it to Database.
Dlgtor, Dlgtee, C-DRM	Revoke(dlgtor,dlgtee, DC, timestamp)	Revocation DC of dlgtee and storing result in Database.
S-DRM	AccessDecision(dlgtee, resource, DC,ACS)	S-DRM makes access decision according to DC.
S-DRM, C-DRM	DeleValiTest(dlgtee, resource, DC, attributes, timeperiod)	S-DRM & C-DRM tests validity of DC periodically through attributes of entities.

Here represents delegation process between agents, Agent_Dlgtor owns some capabilities, Agent_Dlgtee requests a capability for the sake of sharing information and cooperation, material process is as follows:

(1) Agent_Dlgtee calls function DeleRequest (Agent_dlgtor, Agent_dlgtee, resource, capability) on C-DRM to Agent_Dlgtor, and send the request of capability delegation on resource ;

(2) Agent_Dlgtor calls DeleVerify (Agent_dlatee, Agent_dlgtor, capability, ACS) on C-DRM to verify the request;

(3) Agent_Dlgtor calls Delegation (Agent_dlgtor, Agent_dlgtee, DC, timestamp) subscribes Delegation Certification with Agent_Dlgtee on C-DRM, at the same time the DC is sent to delegation database; then DeleReply (Agent_dlgtee, Agent_dlgtor, result) is called on C-DRM to send verification result to Agent_Dlgtee.

(4) When Agent_Dlgtee begin to access to shared information, AccessDecision (Agent_dlgtee, resource, DC, ACS) acts on S-DRM for access decision via delegation database; in the procedure of access, DeleValiTest (Agent_dlgtee, resource, DC, attributes, timeperiod) test delegated capability validity in every given periods by system and delegator.

(5) If the attributes of Agent_Dlgtee,Agent_Dlgtor, or resource change, Server could pause access and C-DRM calls the function Revoke (Agent_dlgtee, Agent_dlgtor,

DC, timestamp) to revoke delegation, then stores result in database.

5. A Practical Application Based on the Model

The designing and realization of Agent-Based Text Information Retrieval System (ABTIRS) adopted this model, so it improved on the issue of agent's capability and cooperative work, advancing system efficiency and parallel processing. The system defined and realized User Interface Agent (UIA), Information Collection Agent (ICA), Information Processing Agent (IPA), Interests Learning Agent (ILA)and their corresponding role based on different responsibility and capability, such as UIAR, ICAR, IPAR and ILAR, respectively accomplishing retrieval demand analyzing and feedback, text information collection, information filtering, as well as user' interests collection. Multi-Agent Delegation Management Subsystem (MADMS) mainly implemented agent-role assignment, role-capability assignment, capability delegation and collaboration constraint. Especially, capability delegation between agent roles was processed based on the above mentioned architecture and key protocol functions in section 4.Because agents are intelligent and independent, they could not only adjust and activate their new roles according to the change of ACS, but also delegate their capabilities or roles to the other agents for sharing resource and cooperation in this system. The framework of ABTIRA is as following figure3.

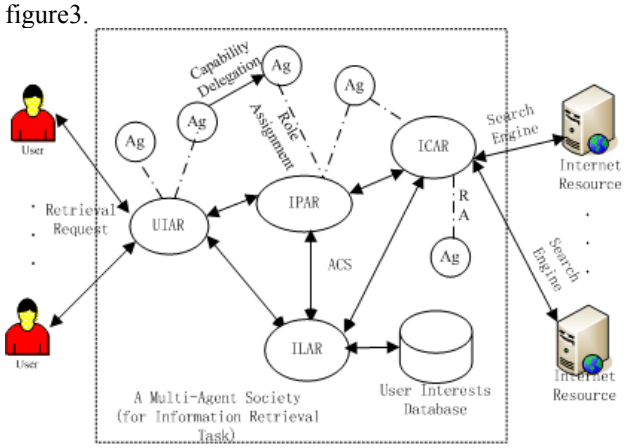

Figure 3. Framework of ABTIRA based on model

6. Conclusions

CDM for MAS is a dynamic multi-agent capability delegation reference model supporting time character and delegation constraint for the sake of collaboration. Based on the model, agent's capability is not single and static, but multiple and dynamic assignment and delegation, at the same time ACS and agent role/capability constraints improve agents' cooperative work and security of MAS. The future research is enriching the model semantics and further formalization, especially in dynamic attributes change and management.

References

[1] J.S. Wang and S.G. Cui, "Multi-Agent Technology and Applications", *Computer Engineering and Applications*, 2003, 39(18), 61-62+66.

[2] M. Ren, and C.D. Wang, "Research of Multi-agent Cooperating Based on Federation Structure", *J. of East China University of Science and Technology*, 2004, 30(3), 311-314.

[3] A. Omicmi, A. Ricci, and M. Viroli, "RBAC for Organisation and Security in an Agent Coordination Infrastructure", *Electronic Notes in Theoretical Computer Science*, 2005 (128), 65-85.

[4] H.X. Lai and S. Zhang, "Role-Based Multi-Agent Workflow Model", *Computer Applications*, 2004, 24(12), 316-318.

[5] H.B. Zhu. "Some issues of Role-Based Collaboration", *Proceedings of IEEE CCECE2003- CCGEI2003*, Montreal, May, 2003.

[6] A.R. Tripathi, T. Ahmed, and R. Kumar, "Specification of Secure Distributed Collaboration System", *Proc. of IEEE the sixth International Symposium on Autonomous Decentralized System*, 2003.

[7] LI Cheng-kai,ZHAN Yong-zhao and XIE Li, "A Role-Based Access Control Model for CSCW Systems", *Journal of Software*, 2000,11(7), 931-937.

[8] D. Xiao, C. Liu, and X. Chen, "The security model of CSCW system based on RBAC", *J. Huazhong Univ. of Sci. &Tech. (Nature Science Edition)*, 2004,32(5), 56-58.

[9] Z.Y. Zhang, J.X. Pu, and C.Y. Feng, "Multi-Agent System Management Model Based on Role & Agent Collaboration Scenarios", *Computer Application and Software,* 2005, 22(10), 107-109.

[10] X.W. Zhang, S.J. Oh, and S. Ravi, "PBDM:A Flexible Delegation Model in RBAC", *Proceedings of SACMAT2003*,June2-3,Como,Italy.

[11] E. Barka, and R. Sandhu, "A Role-Based Delegation Model and Some Extensions", *Proceedings of the 16th Annual Computer*, Sheraton New Orleans, 2000.

[12] S.Y. Na , S.H. Cheon, "Role Delegation in Role-Based Access Control", *Proceedings of ACM RBAC2000*，2000.

[13] Z.Y. Zhang, and J.X. Pu, "Permission-Role Based Delegation Model and Object-Oriented Modeling", *Proceedings of China National Open Distributed and Parallel Computing Symposium 2004*, Beijing, China, Oct.19-21, 2004, pp. 52-55.

[14] Z.Y. Zhang and J. Pu, "Delegation Model for CSCW Based on RBAC Policy and Visual Modeling", Proc. of the 11th Joint International Computer Conf. 2005， World Scientific Press Company, Chongqing, China, Nov.10-12, 2005, pp. 126-130.