

# Encourage Participants' Contributions by Roles

Haibin Zhu, *Senior Member, IEEE*

Department of Computer Science and Mathematics  
Nipissing University, 100 College Drive, North Bay, P1B 8L7, Canada  
[haibinz@nipissingu.ca](mailto:haibinz@nipissingu.ca)

**Abstract** - *Teamwork requires every member to contribute to the team. Encouraging people to contribute in collaboration is exciting and challenging. The model E-CARGO could support collaboration with human users, agents, roles and groups. By assigning roles and transferring roles, we can organize and encourage human users and agents to contribute in collaboration and make the teamwork more profitable.*

*This paper analyses the behaviors of people in collaboration, proposes a methodology with roles to improve collaboration by assigning human users more roles, discusses the basic methods to evaluate and promote human users with roles and relevant properties and describes the implementation of the kernel mechanisms.*

**Keywords:** Role, Role-Based, Contribution, Collaboration.

## 1. Introduction

People cooperate in order to obtain better or more productive results than individuals do and collaboration is inevitable and always present [7, 8, 9, 13]. Collaboration may produce both good and bad consequences [5]. As a tool to support collaboration, a collaborative system should definitely help a group of people to produce good consequences and should provide wealthy, attractive and worthwhile experiences for participants [10]. However, the question of how to maximize productive collaborations by manipulating the configuration of teams is still largely unexplored [18].

Computer-supported collaborative systems are virtual societies created by system developers. The design of such systems should reflect the rationalities in natural society. We believe that roles are good mechanisms in supporting collaborative work with computers [19, 20, 21, 22], because roles really improve the management of groups and collaborative jobs [1, 3, 23] and each role facilitates coordination with groups of people and accomplishment of tasks [16].

The process of collaboration can affect participants' satisfaction, motivation, and productivity. How to design the process of collaboration largely affects the success of the collaboration among a group of people [8]. Many people in collaboration are passive and only act as requested. To encourage people to work more is a general aim for all the regulations or policies for societies or organizations. To build a system that can encourage people to actively contribute is exciting and challenging.

The initiatives of our work are to encourage the participants to contribute as much as possible. The basic

assumptions are based on the Maslow's hierarchy [12] of needs. We can say that diligent people hope to become great members in a great group. Those people that never contribute to a team will be finally discarded. Based on this assertion, we assume that most people would like to improve themselves by actively participating collaboration. We can divide people into different types:

- (1) Diligent people try to do as many jobs as possible. They want take as many responsibilities as possible do not care about rights.
- (2) Ordinary people only want to complete the regular jobs assigned to him. They would like to take responsibilities but require the matching rights.
- (3) Lazy people try to do as less jobs as possible. They want to grasp as many rights as possible and take as few responsibilities as possible.

To restrain type (3), encourage type (1), and support type (2), roles should be composed of both rights and responsibilities. That is to say, more rights mean more responsibilities. If people hope to grasp more rights, they must play more roles. As a result, they must take more responsibilities. For those who do not care about rights, there would be still enough rights for them to take more responsibilities. This is the fundamental principle in our past publications [19, 20, 21].

With this fundamental principle, we propose to encourage the participants' contributions to a group by role mechanisms. By roles, people can express progresses by telling how many significant roles they are playing. That is to say, roles can be taken as a flag of advances for people in collaboration.

In a role-based team, lazy people will be finally restricted and discarded by the team and diligent people will be finally the majorities of the team and be promoted to play more important roles that give them opportunities to contribute more.

By introducing roles as tags or flags to approve or disapprove the behaviors of agents in a group, this paper discusses the continuations and expansions of our work on E-CARGO model [21, 22] and its implementation.

This paper is arranged as follows. Section 2 restates concisely the E-CARGO model for role-based collaboration; section 3 analyses the process of role-based collaboration with promotion requirements; section 4 discusses the behaviors of human users and agents in a role-based system; section 5 explains the major mechanisms to process the promotions for human users and agents by roles; section 6 shows the

implementation of the major procedures. The last section concludes the paper and proposes the issues that require more comprehensive research.

## 2. E-CARGO MODEL

Role design is one of the primary methods for improving the quality of collaboration. Role specification is the deliberate, purposeful planning of positions, including all their structural and social aspects and their effects [8]. Our model E-CARGO has contributed to this work.

In our model E-CARGO [21, 22], a collaborative system  $\Sigma$  can be described as a 9-tuple  $\Sigma ::= \langle C, O, \mathcal{A}, \mathcal{M}, \mathcal{R}, \mathcal{E}, \mathcal{G}, s_0, \mathcal{H} \rangle$ , where

- $C$  is a set of classes;
- $O$  is a set of objects;
- $\mathcal{A}$  is a set of agents;
- $\mathcal{M}$  is a set of messages;
- $\mathcal{R}$  is a set of roles;
- $\mathcal{E}$  is a set of environments;
- $\mathcal{G}$  is a set of groups;
- $s_0$  is the initial state of a collaborative system; and
- $\mathcal{H}$  is a set of users.

To encourage contributions of collaborators, we emphasize the design of agents, roles and messages.

An *agent* is a special object that represents a user involved in collaboration. It is defined as  $a ::= \langle n, c_a, s, \mathcal{N}_r, \mathcal{N}_g \rangle$ , where

- $c_a$  is a special class that describes the common properties of users;
- $n$  is the identification or name of the agent;
- $s$  is the set of properties of the agent;
- $\mathcal{N}_r$  means a set of identifications of roles the agent are playing; and
- $\mathcal{N}_g$  means a set of identifications of groups the agent belongs to.

A message is defined as  $m ::= \langle n, v, l, \mathcal{P}, t \rangle$  where

- $n$  is the identification of the message;
- $v$  is null or the receiver of the message expressed by an identification of a role;
- $l$  is the pattern of a message, specifying the types, sequence and number of parameters;
- $\mathcal{P}$  is a set of objects taken as parameters with the message pattern  $l$ , where  $\mathcal{P} \subset O$ ; and
- $t$  is a tag that expresses any, some or all message.

A *role* shows both the rights and responsibilities human users. We use incoming messages to express responsibilities and outgoing messages rights. It is defined as  $r ::= \langle n, I, \mathcal{N}_a, \mathcal{N}_o \rangle$  where,

- $n$  is the identification of the role;
- $I ::= \langle \mathcal{M}_{in}, \mathcal{M}_{out} \rangle$  denotes a set of messages, wherein,  $\mathcal{M}_{in}$  expresses the incoming messages to the relevant agents,  $\mathcal{M}_{out}$  expresses a set of outgoing messages or message templates to roles, i.e.,  $\mathcal{M}_{in}, \mathcal{M}_{out} \subset \mathcal{M}$ ;

- $\mathcal{N}_a$  is a set of identifications of agents that are playing this role; and
- $\mathcal{N}_o$  is a set of identifications of objects including classes, environments, roles, and groups that can be accessed by the agents playing this role.

In such a system, we can view a human user and his/her agent as a whole entity, i.e.,  $\mathcal{A}$  and  $\mathcal{H}$  are tightly-coupled sets. We can say that a human user and his/her agent play a role together. With this tight coupling, we emphasize that a role-based collaborative system is composed of both computers and human beings. When a human user provides more implementations for the methods to respond the incoming messages, the agent can do more jobs. Gradually, agents advance towards to autonomous agents [4]. With the system's help, people could collaborate to accomplish specific tasks. This model could be used to express a natural evolution of a man-machine (intelligent) system. In the following discussions, an agent represents the agent and the human user it represents.

## 3. THE PROCESS OF ROLE-BASED COLLABORATION

To understand the process of role-based collaboration, we need to clarify the categories of collaboration. Besides the categorization based on places and times [14], we can have another method to categorize different collaboration.

Collaboration can be divided into to types based on collaboration times:

- Synchronous collaboration: it is face-to-face collaboration or virtual face-to-face collaboration with the help of collaborative systems.
- Asynchronous collaboration: it occurs at different times at ordinarily different places with the help of paper-based or computer-based communications.

Collaboration can be also divided into two categories based on collaboration periods:

- Long-term collaboration: collaboration takes a long time, such as a large project or a large group such as an enterprise.
- Short-term collaboration: it takes a short time such as a meeting and a conference.

Synchronous collaboration normally supports short-term collaboration. Asynchronous collaboration prefers to long-term collaboration. Evidently, E-CARGO prefers to support asynchronous and long-term collaborations. But it can also be used to support short-term synchronous collaboration. Based on the assumptions for collaboration:

- Any person has a limited ability. If s/he plays too many roles, there must be a chance that s/he cannot serve all the requests satisfactorily that will affect his/her credits.
- A person can collect his/her credits by serving others.
- To assign new roles is to appraise the person's past work.

- A role might be disapproved based on the credit of the person to take responsibilities.
- The role dispatchers are fair and will distribute messages evenly.

We can describe the procedure of role-based collaboration as follows which introduces promotions for agents [22]:

- Step 1: Negotiate roles. Agents discuss or negotiate to specify the roles relevant to collaboration. If no agreement is reached, the collaboration aborts. There are two ways to negotiate roles. One is initiated by an agent and the other is initiated by a role facilitator. The former is that an agent wants to play a role and the latter is that the role facilitator wants to nominate an agent to play a role.
- Step 2: Assign roles. Every agent is assigned one or more roles. If no agreement is reached, the collaboration aborts. Both the above negotiation ways will be finally the same process, i.e., the human user should provide a class for his/her agent to cover all the services required by the role. In reality, there might be declarations or working plans for a person to play a role.
- Step 3: Play roles. Agents work according to their roles until collaboration is completed successfully or satisfied to promotion.
  - Step 3.1: Check incoming messages. Agents understand what they need to do at this time. The incoming messages are confined by the role responsibilities (the service interface). The agents collect credits by serving others. If there are conflicts, discontents or promotion requests, go to Step 1.
  - Step 3.2: Issue outgoing messages. Agents need to access and interact with the system by sending messages, or asking for others' services in order to provide their services. If there are no incoming messages, agents may issue messages as they want. The messages are confined by their roles' rights. If there are conflicts, discontents and promotion requests, go to step 1.

The general activities in a role-based system can be listed as follows:

- To play a role. An agent wants to grasp rights and to take responsibilities.
- To serve others. An agent responds an incoming message.
- To evaluate an agent. The system evaluates an agent by checking the information relevant to it if a role is appropriate to it.
- To disapprove a role. The system takes away a role from an agent.
- To promote an agent. This system upgrades an agent by assigning it a super role.

In order to facilitate the above activities, we need to consider several important factors as follows:

- How to collect the credits for an agent? We can set a weight number for a message. While an agent has served

this message, this weight is added to the credits of the agent.

- How to dispatch messages? The role dispatcher should have no bias. There should be a procedure to balance the dispatching, such as considering the factors of the credits, the abilities, and the working state of an agent.
- How to check if an agent is really capable of doing what the role requires? The only way is by the agent's declarations of abilities and credits.

## 4. HUMAN USERS AND AGENTS

Logically, we can consider an agent and the human user it represents as the same entity. However, to build a role-based system, we still need to consider agents and human users separately. In a system, we can attach roles to an agent to express a human participant's playing this role. When human users use the system, they contact others through roles.

When using role-based systems, human users improve their agents to accomplish more and more services and they will obtain more rights based on more roles. The ordinary tasks for them are as follows:

- Task 1: provide services. Make a class of agent that includes all the implementations of the methods corresponding to the incoming messages of the roles.
- Task 2: ask for services. To send messages to roles and obtain services from other agents.
- Task 3: accessing objects. To obtain the properties of objects constrained by the roles.
- Task 4: request to promote. To play more roles.

Task 1 is the most important task for a human user. At least, a human user could provide a class that implements in the simplest forms (for example, return true) all the required services by the role. Tasks 2-4 can be done with the operations from human users and they actually help human users accomplish task 1 more efficiently.

Based on our E-CARGO model, an agent is composed with five elements, i.e.,  $n$ ,  $c_a$ ,  $s$ ,  $\mathcal{N}_r$  and  $\mathcal{N}_g$ . We can provide special states and special methods in the class  $c_a$  to facilitate the requirements for agents discussed in section 1. To manage an agent's progresses, we need to introduce abilities, credits, and workload.

We use  $\langle e_t, e_s \rangle$  to express the ability for an agent, where  $e_t$  expresses how many units of free time it has and  $e_s$  expresses how many units of space it has.  $\langle e_t, e_s \rangle$  can be reset based on the performance of an agent's services. Even though the time in one day is 24 hours, a human user may have different  $e_t$  and  $e_s$  based on their abilities including the working efficiencies, attitudes, and goals.

When a new agent is created, its ability can be initialized with the declaration of its relevant human user. This declaration is difficult to evaluate as a cheat or a wrong appraisal. But any way, a cheat or a wrong appraisal would lead to bad affects to the agent and its human users. There

might be conservative values or aggressive values for  $\langle e_b, e_s \rangle$ , the former lower the actual value the human user can offer and the latter upper the actual value. These values could be adjusted by the human users and the role facilitators.

We use a credit  $w$  to express the past performance or credits of serving others. It can be used as a criterion to assign roles. The difference between abilities and credits is that the ability is declared by the agent and the credit is earned by an agent's past working.

Based on our assumption, an agent hopes to play as many roles as possible. Similarly, it must try to complete as many services as possible to collect credits in order to play more roles. To avoid an agent has too much workload to accomplish services efficiently. Each agent has a busy state  $u$  to express its workload. The busy state can be set by role facilitators or by human users. Therefore, an agent can be redefined as  $a ::= \langle n, c_a, s, N_r, N_g, e_b, e_s, w, u \rangle$ , where  $n, c_a, s, N_r$ , and  $N_g$  keep the original meanings,  $e_b$  and  $e_s$  are used to express the ability,  $w$  the credit and  $u$  the workload.

To collect service credits, a big problem of an agent is how to reply the service. One method is called synchronous reply, i.e., to reply directly to the role within a method in the agent class. The other is called asynchronous reply, i.e., to reply by human users' operations to the role. This method is common but difficult to implement. The role class should be able to record the message transferring state, match the reply with the early request, and let agents collect credits.

Another problem is how to evaluate the performance of a human user and how to avoid cheating. It might be easy for a human user to cheat in a role-based system. For example, s/he may only provide a simple reply method implementation for an incoming message to collect the credit relevant to the service without doing actual work.

To deal with this problem, we need to learn from our daily lives. From daily lives, we know that this will be found by the following work of other human users, because if the former request was satisfied, the following request should be well done. If it cannot be done as supposed, it means there is something wrong with the former work. This cheating will finally lead to that the human user is abandoned from collaboration. Role facilitators will be able to judge and reset the credits of some agents.

## 5. ROLES

In the model of E-CARGO, a *role* has four elements as its properties, i.e.,  $n, I, N_a, N_o$ . We can enhance a role by adding some methods in the role class to evaluate an agent and decide if the agent can play a role. In our method, we view playing a new role as a promotion for an agent and its human user. Some authors take it as evolution [2, 6, 17, 11].

The role entity should provide a method to attach an agent to a role and a method to find an agent to execute the message.

These methods affect the promotion of the human users, i.e., to play more roles.

The former method should check if the agent can be attached to the role and return true or false as a result. The latter one should dispatch the message evenly and without bias.

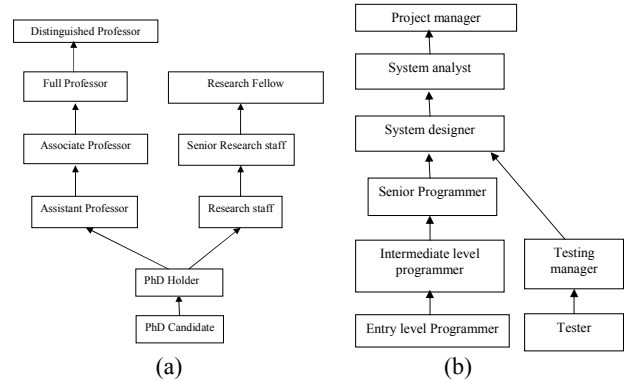
To determine if an agent is qualified to play a role, there are several criteria:

- The agent must provide methods to cover all the incoming messages for the role;
- The agent must collect enough credits;
- The agent must have enough abilities;
- At least one role the agent is playing belongs to the subordinate roles of this role if the role has subordinate roles; and
- At last, when a role is approved for an agent, the ability  $\langle e_b, e_s \rangle$  of the agent should be decreased based the role's  $\langle e_b, e_s \rangle$  that expresses the ability requirement to play this role.

To manage credits, we assign a weight for each incoming message of a role. When we specify a role, we can assign incoming messages to a role and set the specific weight for the role. That is to say, the same incoming messages might have different weights for different roles. When an agent executes a method relevant to an incoming message, it will collect the weight of the message to its credits.

A message is redefined as  $m ::= \langle n, v, l, P, t, w \rangle$  where

- $n, v, l, P$ , and  $t$  keep the original meanings; and
- $w$  is the weight for an agent to collect its credit for promotion, i.e., if an agent or its human user responds this message the agent will get the weight and accumulate it to its credit.



**Fig. 1 The role hierarchies: super and subordinate relationship (a)The role hierarch for a PhD candidate; (b)The role hierarchy for the members in a software development team.**

To facilitate role playing, roles should construct a hierarchy (Fig. 1). A role hierarchy is actually a partial order on roles  $\mathcal{R}$  and a relation  $>$ , i.e.,  $\langle \mathcal{R}, > \rangle$ .  $\forall r_1, r_2 \in \mathcal{R}, r_1 > r_2$  means that  $r_2$  is a subordinate role of  $r_1$ . To express this hierarchy, we need an item in the role entity to accommodate the identifications of subordinate roles and super roles. Note that this relationship expresses the promotion direction for roles. It is different from the role inheritance relationships described by Gottlob et. al. [6].

For example, a PhD candidate role has two career ladders for promotions shown in Fig. 1. From Fig. 1, we find that there are partial orders among groups of roles.

At this step, we know that a role should be redefined as  $r ::= \langle n, I, N_a, N_o, e_t, e_s, R_m, R_s, w \rangle$  where,  $n, I, N_a, N_o$  keep the original meanings,  $e_t$  and  $e_s$  are used to express the ability requirement,  $R_m$  the super roles,  $R_s$  the subordinate roles, and  $w$  the credit requirement.

There are two kinds of promotions for human users or agents: one is to assign new roles to them and another one is to enlarge the roles human users and their agents are playing [8]. E-CARGO can accommodate both ways. In reality, both ways will be dealt with one way, i.e., changing roles for agents. If a role is enlarged, it will be different with the original one. Suppose there more than one human users are playing it, we must create a new role for the special promotion. That is to say, we need to create a new role that is the enlargement of the original role and assign it to the agent whose human user's role needs to be enlarged.

To assign (nominate) a role to an agent or a human user is done by a human user with a special role. Before the human user accepts this assignment (nomination), the role is in the state of nomination for the agent. The human user can choose between accepting and rejecting the role assigned. To accept the role, the human user must provide enough methods to play this role. The role entity again checks if it is qualified.

## 6. IMPLEMENTATION

Based on our implementation of the kernel mechanisms of our model E-CARGO [21, 22], we redesigned the classes C\_Role, C\_Agent and InMessage, such as adding new member variables credit, T and S to C\_Agent, adding credit, subordinates and supers, T and S to C\_Role, and adding weight to InMessage.

```
class C_Agent
{
    //...
    int credit; //It is used to express its credits to provide services
    int T; //the number of time units (speed)
    int S; //the number of space units (memory)
    //...
}
class C_Role
{
    //...
    C_Role [] subordinates;
    //An array of roles as its subordinate roles
    C_Role [] supers; //An array of roles as its super roles
    int T; //the number of time units used to play this role
    int S; //the number of space units used to play this role
    int credit; //the credit number to play this role
    //..
}
class InMessage
{
    public Method aMethod; //A Java method
    public int weight; //It expresses the credit weight.
    //...
}
```

In class C-Role, the key methods “addAnAgent” and “dispatch” are also greatly modified as follows.

```
public boolean addAnAgent(C_Agent a)
//approve the agent to play this role
{
    //1. Check if the agent has enough services
    Class c = a.getClass();
    Method [] InMessages = c.getMethods();
    for (int i = 0; i < in_size; i++)
    {
        for (int j = 0; j < InMessages_num; j++)
        if (incomings[i] != null)
            {if (incomings[i].equals(InMessages[j]))
                {
                    break;
                    //check if the message is in the InMessages of
                    //the class of agent a
                }
            }
        return false;
    }
    //2. Check if the agent collects enough credits
    if (a.credit < credit ) return false;
    //3. Check if the agent has enough abilities (<T, S>)
    if ((a.T < T) || (a.S < S)) return false;
    else { //It means playing a role occupies time and space of
        //an agent.
        a.T -= T;
        a.S -= S;
    }
    //4. Check if there are subordinate roles for this role,
    //if yes, check if the agent has played the subordinate role.
    if (subordinates_num != 0)
    {
        boolean tag = false;
        for (int i = 0; i < subordinates_num; i++)
            for (int j = 0; j < a.current_roles_num; j++)
                if (subordinates[i] == a.current_roles[j])
                    { tag = true;
                      break;
                    }
        if (!tag) return false;
    }
    //5. Link the role to the agent
    if (agents == null)
    {
        agents = new C_Agent [agent_limit];
        agent_index = 0;
        agents[agent_index++] = a;
    }
    else
    {
        agents[agent_index++] = a;
    }
    return true;
}
public void dispatch(OutMessage aMess)
//send message aMess to the agents playing this role
{
    int wght;
    if ((wght = search(aMess)) != 0)
    //Check if the message belongs to the role.
    {
        if (aMess.type == "all") //all messages
            for (int i = 0; i < agent_index; i++)
                //dispatch the message to the agent
                agents[i].accept(aMess, wght);
            }
        else if (aMess.type == "any") //any message
            {int index = 0;
             int max_credit = 0;
             for (int i = 0; i < agent_index; i++)
```

