

Role-Based Chatting

Haibin Zhu

Department of Computer Science and Mathematics, Nipissing University, North Bay, ON, Canada

Email: haibinz@nipissingu.ca

Rob Alkins

Department of Computer Science and Mathematics, Nipissing University, North Bay, ON, Canada

Email: alkins.rob@gmail.com

Matthew Grenier

Department of Computer Science and Mathematics, Nipissing University, North Bay, ON, Canada

Email: matty_g07@hotmail.com

Abstract— Role-based collaboration (RBC) is proposed to support improved human collaboration through more usable human-computer interfaces. To satisfy the objectives of RBC, new practical tools are required.

Role-based chatting is a typical instance of role-based collaboration. To design a role-based chatting tool involves almost all the fundamental problems of RBC. This paper presents the design problems of role-based collaboration including model, engine and interface design, illustrates the scenario of role-based chatting, and describes the implementation of a role-based chatting tool including its architecture and components. The design of the chatting tool reflects all the principles outlined by role-based collaboration. This tool shows that role-based collaboration is practical and feasible. It also shows the possibility of building more complex role-based systems.

Index Terms—Roles, role-based, chatting, role-based collaboration

I. INTRODUCTION

Computers are globally pervasive and computer-based collaboration is becoming an expected user skill. Now, most people complete their daily routine tasks with computers. Emailing and surfing the Internet are now common daily activities. However, there are still many people who do not like using computers and resist using the technology if at all possible. This challenges computer scientists and engineers to create more usable software and hardware products. If computer-based tools can be seen as easy to use and productive, user reluctance can be overcome.

Computer-based tools should facilitate the completion of daily activities. They should not only support real/virtual face-to-face collaborative environments but also improve such collaboration by providing mechanisms for overcoming certain drawbacks such as the manipulation of a meeting by an aggressive person [19, 22].

On line chatting is pervasive due to the development of the Internet. Young people spend considerable time on this activity. Chatting rooms provide a useful platform for collaboration, learning and enjoyment. Designing and

improving the chatting rooms and tools for chatting are required. Role-based chatting is one exciting way to improve current on-line chatting tools.

Role-Based Collaboration (RBC) [21] is a methodology to design and implement computer-based systems. It is an approach that can be used to integrate the theory of roles into Computer-Supported Cooperative Work (CSCW) systems and other computer-based systems. It consists of a set of concepts, principles, mechanisms and methods. The properties of RBC are as follows [21]:

- Clear role specification: it is easy for human users to specify and understand their responsibilities and rights.
- Flexible role transition: it is flexible and easy for a human user to transfer from one role to another.
- Flexible role facilitation: it is easy for role facilitators to modify roles. Because collaborative activities are constantly evolving, even the existing roles might be required to adjust in correspondence with the development of the system.
- Flexible role negotiation: it is easy to negotiate a role's specification between a human user and a role facilitator.
- The interactions among collaborators are through roles.

RBC imposes challenges and benefits not found in traditional CSCW systems [22]. Role-based collaboration can help people in both long-term and short-term collaboration. It will provide benefits as follows:

- In long-term collaboration, roles help
- Identify the human user "self" [5];
 - Avoid interruption and conflicts [8, 15];
 - Enforce independency by hiding people under roles [1];
 - Encourage people to contribute more [9, 17]; and
 - Remove ambiguities to overcome expectation conflicts [2, 6, 21].

In short-term collaboration, roles help

- Work with personalized user interfaces [13];

- Concentrate on a job and decrease possibilities of conflicts for shared resources [21];
 - Improve client's satisfactions with more people playing the same role during a period; and
 - Transfer roles with the requirement of a group [23].
- In management and administration, roles help
- Decrease the workload of system administrators;
 - Separate of concerns in designing;
 - Decrease the knowledge space of searching;
 - Create dynamics for components; and
 - Regulate ways of collaboration among agents.

From the above benefits, RBC research will bring exciting improvements to the development and the application of CSCW systems, and the methodologies of collaboration. There are, however, requirements for tools to facilitate roles and interaction among roles.

Collaboration occurs in different forms. There is no doubt that chatting is a common form of collaboration. Role-based chatting is a special case of RBC and investigating role-based chatting will unveil most of the mysteries of RBC.

This paper is arranged as follows: Section II discusses the role-based chatting including its scenario and benefits; Section III clarifies the design problems of RBC; Section IV presents the implementation of a tool for role-based chatting; Section V reviews the related work; and Section VI concludes the paper and proposes topics for future research.

II. ROLE-BASED CHATTING

In role-based chatting, every person plays one or more roles and the people involved do not have to know each other (Figure 1). Based on [21], role-based collaboration is facilitated by role-specification, role-negotiation, role-assignments, and role playing.

The scenario of role-based chatting is as follows:

- 1) A chatting room (group) is built with roles related to a domain.
- 2) The roles in the chatting room are clearly specified and presented.
- 3) People login the room, check the roles, and decide to serve or be served.
- 4) If people want to serve, they select roles they want

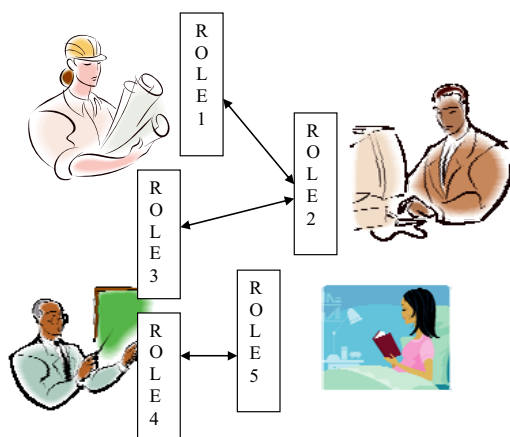


Figure 1. Role-based chatting.

to play and prepare to answer questions.

- 5) If people want to be served, they select roles to ask questions.
- 6) In the chatting tool [20], a credit is set for each person in his or her agent.
- 7) The people in chatting are evaluated with credits by the system based on their performance in playing roles.
- 8) The people's intent to play roles is restricted by his or her credits.

In reality, client service is widely used in technical services of a company. Such a service is generally facilitated by phone or email. For clients, this service is somewhat "role-based", because they do not care who answers their questions, even though the person who answers the technical questions begins by telling the client his/her name, in fact, first name. A client's primary concern is service provider qualification.

In role-based chatting, we also emphasize that the people involved do not necessarily care about who they are talking with. From a psychological point of view, this encourages shy people to participate in chatting, expressing their real opinions, collaborating, helping and seeking help.

There may be arguments in favour of naming the participants. *Specifying names can encourage service providers to work more effectively.* In fact, through role-based chatting, service providers can be motivated by the fact that managers can attach names to service providers when role scheduling. *Current internet-based chatting is anonymous. Anonymity and nicknames on the Internet protect people from being known by their real names.* However, anonymity significantly degrades the quality of suggestions, recommendations, and decisions, avoiding the issue of technical/professional qualification. In such an anonymous situation, the value of chatting or collaborating may be in doubt due to the potential for participants to be misleading. Anonymous collaborators have no responsibility to guarantee that what they have to say is true. Role-based chatting could overcome these drawbacks based on role definitions in the system.

This kind of chatting tool [20] helps form a positive community that encourages people to help each other without necessarily knowing each other. Eventually, although they never know each other, they are treated fairly by the system. People seeking service through chatting may be comforted by knowing that providers are well suited to the task.

Therefore, role-based chatting has the benefits in service management and collaboration as follows:

- 1) It is easy for a manager to distribute tasks because it is easier to find a technician than to find a specific person.
- 2) Managers have ways to evaluate their staff.
- 3) Role-based chatting provides a balanced way for anonymity and credibility.
- 4) Two or more people can play the same role to chat with another person to improve the efficiency of a specific service and make the client feel more comfortable and more satisfactory.

- 5) One person can play different roles and serve many clients at the same time period to save the human resources of a company.
- 6) Role-based chatting can encourage people's participation by specially designing some facilities such as credit [9, 17].

III. DESIGN PROBLEMS OF RBC

A. E-CARGO Model

The presented tool [20] is actually a simplified instance of our E-CARGO model [21] where *roles* represent a title with both responsibilities and permissions, and *agents* represent human users of the tool.

- A role is defined as $r ::= \langle n, I, \mathcal{A}_c, \mathcal{A}_p, \mathcal{A}_o, \mathcal{R}_x, O_r \rangle$, where
- n is the identification of the role;
 - $I ::= \langle \mathcal{M}_{in}, \mathcal{M}_{out} \rangle$ denotes a set of messages, where \mathcal{M}_{in} expresses the incoming messages to the relevant agents, and \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{A}_c is a set of agents who are currently playing this role;
 - \mathcal{A}_p is a set of agents who have the potential to play this role;
 - \mathcal{A}_o is a set of agents who previously played this role;
 - \mathcal{R}_x is a set of roles interrelated with r ; and
 - O_r is a set of objects that can be accessed by the agent playing this role.

The above definition indicates fundamental requirements for managing roles and presenting them. For example, $I (\langle \mathcal{M}_{in}, \mathcal{M}_{out} \rangle)$ is definitely used in the presentation of a role. Others must be managed in the system and some of them should be presented with the requirement of collaborative tasks. For a chatting user, it is not required for them to know the internal structure of the roles s/he is playing. However, for an administrative user, it is required for the interface to show the internal structure of a role and the role's relationships with other roles.

An *agent* is defined as $a ::= \langle n, c_a, s, d, r_c, \mathcal{R}_p, \mathcal{R}_o, \mathcal{G}_a \rangle$, where

- n is the identification of the agent;
- c_a is a special class that describes the common properties of users;
- s is the qualifications of the agent;
- d is the credit of the agent;
- r_c is a role that the agent is currently playing. If it is empty, then this agent is free;
- \mathcal{R}_p is a set of roles that the agent can potentially play ($r_c \notin a.\mathcal{R}_p$); and
- \mathcal{R}_o is a set of roles that the agent played previously; and
- \mathcal{G}_a is a set of groups that the agent belongs to.

In this definition, the credit d is taken as an attribute of an agent. The credit d is a special object that can be

seen by its agent but cannot be modified by the agent. It is unusual in software engineering for an object to be prevented from modifying its own attributes. This may be seen as being in conflict with the encapsulation principle of object-orientation. However, it is acceptable in collaboration and interaction. In reality, a person seeking employment is usually more successful when possessing the appropriate certification. The holder is not allowed to modify the certificate. This at-hand certification saves a lot of effort when searching for a qualified individual. In role-based chatting, an agent is considered to represent a human user. One human user corresponds to one agent. In the following discussion, agents, people, and human users are used interchangeably.

A message is defined as $m ::= \langle n, v, d, d_r, l, t \rangle$, where

- n is the identification of the message;
- v is the receiver role of the message;
- d and d_r express the sender agent and senders' role;
- l is the message text by the user; and
- t is a tag that expresses *any*, *some* or *all* message.

In E-CARGO, human users form a group by playing roles in an environment. An environment is defined as $e ::= \langle n, \mathcal{B} \rangle$, where

- n is the identification of the environment; and
- \mathcal{B} is a set of tuples of role, number range and an object set, $\mathcal{B} = \{ \langle r, q, O_e \rangle \}$. The number range q tells how many users may play this role in this environment and q is expressed by (l, u) . For example, q might be (1, 1), (2, 2), (1, 10), (3, 50), It states that how many agents may play the same role r in the group. Where l represents the minimum number of agents required and u represents the maximum number allowed. The object O_e expresses the objects shared by the agents who play the relevant role.

Compared with the O_r of r , O_e is a shared resource in an environment and O_r is a resource only occupied by the role r . The r in an e should be an instance of role, that is to say, the O_r should be instantiated in an e .

A group is defined as $g = \langle n, e, \mathcal{J} \rangle$, where

- n is the identification of the group;
- e is an environment for the group to work; and
- \mathcal{J} is a set of tuples of identifications of an agent and role, i.e., $\mathcal{J} = \{ \langle a, r \rangle \mid \exists r, q, O_e \exists \langle r, q, o \rangle \in e.\mathcal{B} \}$.

In role-based chatting, a chatting room is an environment. People can enter a chatting room, play a role in the room, and construct a group.

B. Role Engine

A role engine can be understood in the same way as a Prolog inference machine. For example, to use a Prolog system, people only need to write rules and facts. The Prolog inference machine will search the result. Similarly, to implement role-based collaboration, based on the proposed role engine, people simply need to specify roles and create agents based on role specifications. When

agents are put into the role engine, the engine will drive agents' work properly to obtain their goals by collaborating with other agents.

A role engine should do the following:

- Manage roles (create, delete, and modify);
- Manage agents (create, delete, and modify);
- Manage the credits (d) of agents;
- Assign roles to agents;
- Build role relations; and
- Check the consistency of the system.

A role engine is a platform for agents to collaborate. On this platform, agents work for the system by playing roles. Through implementation of the role engine, all agents are driven to contribute and work diligently for a system that offers a solution to a real-life problem. A role engine should possess role dynamics, facilitate role transfer, and support role assignments, interaction and presentation.

In the presented role-based chatting tool [20], the role engine is mainly responsible for adjusting the credits of agents and presenting suggestions to human users.

C. Role Interaction

With a role engine, interaction and collaboration among people are facilitated by roles. The role engine controls the messages exchanged among roles. Based on our E-CARGO model [21], interaction is implemented by issuing messages. Dispatching messages to agents or people is a highly intelligent task to be accomplished by a role engine and its roles. It needs to consider several properties:

- Fairness: the engine should dispatch messages evenly to peer agents. It should avoid starvation or overloading, where starvation means that an agent has not received messages for a time longer than a limit while overloading means that an agent receives too many messages in a limited time.
- Consistency: the engine should check the consistency of role hierarchies. When a new role hierarchy is added, the system should be kept consistent.
- Completeness: periodically, the engine should dispatch a specific message to all agents of the targeted group.

In role-based chatting, the above functions can be performed by the administrator in charge of a chatting room. It is the same situation as a manager and his or her client service staff.

D. Role Assignment

In the business world, a person is required to have additional knowledge, skills and habits to be qualified for a new position. Successfully finding a new job is dependent on similarities between new roles and those previously performed by a person [3], i.e., qualifications are the basic requirements for possible role-related activities.

Role assignment is the first event for RBC and is dependent on the qualifications of agents. There are two difficult aspects: when roles are defined as abstract

interfaces, it is necessary to match the abstract interfaces with the concrete things (syntactically and semantically) an agent can do; when roles are specified as concrete processes, it is necessary to specify all the details of the roles exactly in both syntax and semantics and roles should be able to adapt to different agents of the system.

To address role assignment problems, it is necessary to deal with issues arising from role definition and specification. Clear role definition and specification helps a person collaborate by avoiding ambiguities and conflicts [4]. Nobody likes to work in a group lacking clear regulations and rules. A highly efficient, productive society is well-organized, well-regulated and well-managed. Specifications are the key points. Initial questions such as "how is a role defined?" and "how are roles specified?" need to be answered.

After roles are well defined and specified, role assignment can be made based on these specifications and mining methodologies that extract formal specifications from files of natural language related to the qualifications, responsibilities, and rights of agents. Role assignment can also be applied in knowledge extraction from vast amounts of versatile raw data in the grid computing world.

In RBC, agent role matching is a fundamental yet difficult problem. Assignment of an agent to a role requires prior evaluation of the agent's qualifications. However, "qualification" is a complicated criterion comprised of many aspects. In a role engine, a role specification methodology is taken as the solution to the problem of appropriate agent qualification and evaluation.

Any specification language or tool encounters an initial problem, that is, how to strictly express the semantics with simple syntax. For human users, simple syntax facilitates using the tool. In reality, ambiguous language may cause problems anywhere at anytime. It requires strict semantics matching to check whether two objects are the same or not.

Because the assignment of roles to agents is dynamic in a collaborative system, i.e., the roles can be re-assigned to other agents based on the changing of the environment, such assignment needs to consider the following properties:

- Fairness: the engine should assign roles to agents based on a fair rule to avoid starvation or overloading.
- Service quality: the engine should assign roles to qualified agents according to the performance of agents. Special roles may also require many agents to play to save time.
- Least conflict: the engine should guarantee that the roles assigned to agents create the least opportunity for agents to have conflicts in sharing information when they are playing roles.

E. Role Dynamics

Everything exists in the world for a special reason. Every activity in the world is instigated for some reason. These reasons form driving forces for the creation of new things and actions. In society, people in an organization




with good dynamics will work actively and collaboratively toward the common goal of the organization. In contrast, people in an organization without good dynamics cannot work as effectively and may not have a clear understanding of the goals necessary to make the organization competitive.

Collaborative systems should encourage diversity of behavior in a group of people [18]. People join a group because they hope to provide assistance or obtain assistance from others. They may hope to establish a positive reputation through collaboration. This is one kind of dynamics. With well-built dynamics, people will automatically follow the regulations of an RBC system and collaborate with each other to achieve common goals.

F. Role Presentation

Roles are finally presented to users and should be easily understood. The specification of roles should consider the easy implementation of role presentation. Role presentation should consider aesthetics, intuition, and other human factors. Similar to other human user interface requirements, role presentation has the following requirements: Easy to understand; Easy to remember; Used to support personalized user interface; and Presented in a multi-media style, such as text, image, audio, video and animation presentations.

To meet the requirements as above, iconizing different roles are beneficial to role presentation. Some concrete roles can be easily expressed with icons such as a police

officer , a waiter , and a worker . However, it is difficult to express a computer professor, a software developer, and a system analyst. It is also very difficult to express a generalized concept, like a “role”, by an icon. Even further, it is a very difficult task to design icons presenting as much information as that in a role specification. Therefore, tables, lists, and graphs are needed to present roles.

Evidently, it is easy to express an agent by a human icon and every one knows it expresses an agent or a person. In the design of this tool [20], the biggest problem is choosing an icon to represent a role. A role turns out to be a very general and abstract concept. Therefore, we choose 15 icons from the clip art of the Microsoft Word, and composed a survey question as shown in Figure 2.

In this survey, we issued the question to more than 200 students and faculty at Nipissing University and received 182 valid responses. The data is shown in Table 1. The interesting result is that the “Role (10)” icon is mostly preferred (34/182). The “Mask (13)” icon is the second

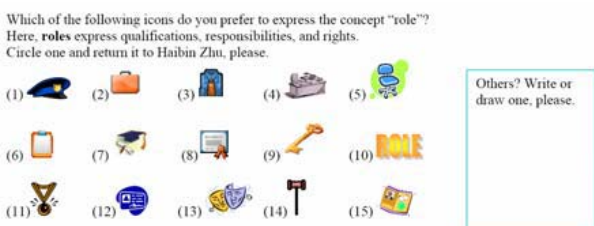

















Figure 2. The main user interface.

TABLE I.
THE SURVEY RESULT FOR THE ROLE ICON PREFERENCES

Icons	Number of votes	Icons	Number of votes	Icons	Number of votes
(1) 	20	(6) 	4	(11) 	4
(2) 	2	(7) 	17	(12) 	5
(3) 	9	(8) 	14	(13) 	28
(4) 	6	(9) 	13	(14) 	12
(5) 	3	(10) 	34	(15) 	3
				Others	8
				Total	182

(28/182) and the “Police cap (1)” icon the third (20/182).

It is observed that the human users’ preferences are diverse. A better way is to provide a list of icons for a user to choose when the system is installed or started. On the other hand, it demonstrates that an abstract concept is difficult to iconize and needs the designers’ creation, imagination and analysis as well as the usability study from the users’ perspectives. A survey can help in the design and it is also a good research topic to express an abstract concept with an icon. The most preferred icon is the “ROLE” icon. To comply with this preference, a similar “R” icon is designed shown in Figure 6.

In the role-based chatting tool, we present the content of a role by a text window to list all the incoming and outgoing messages. The roles’ relationships with other components such as agents and objects are processed internally and presented to a special role, i.e., the administrator.

IV. IMPLEMENTATION

A. The Architecture

As discussed in [19-21], a role-based collaborative system should be built on client/server architecture. In this tool, most of the foundation concepts discussed in Section III, such as, role, agent, message, permissions and responsibilities, are designed as classes and the design of server and client is based on these foundation classes. The tool is composed of a server, many clients and an SQL server (Figure 3).

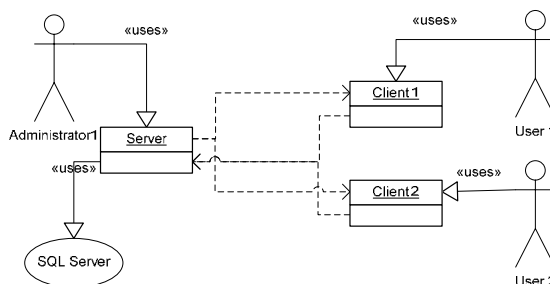


Figure 3. The architecture of the tool.

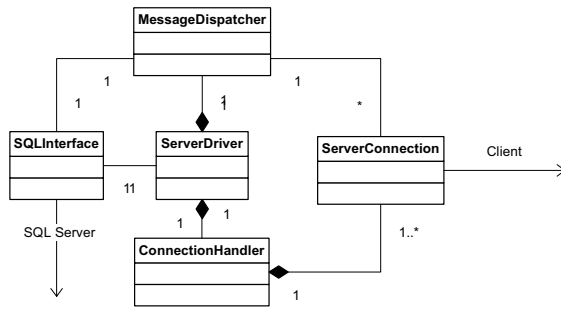


Figure 4. The server side classes.

The server provides the management of information and controls the interactions among client users. It sends and receives information and requests, and makes calls to SQL server in order to manage all the information of the system including roles and agents (Figure 4).

Based on the client/server architecture, all the information in the system is stored in an SQL server database with associated date and time entries. In this way, at any given moment, the state of the system is completely specified by the information on the SQL server which is installed in a powerful server computer. Using this approach, there is no need to save the system information upon the shutdown of a client, nor is there any critical data that may be lost due to a power failure of a client. All the updates to the system are preformed in a single database command. For example, when a user sends a message, the message is first stored in a database table entitled "post office", it remains there until the MessageDispatcher thread obtains appropriate recipients, only then is the data moved to the recipients' mailboxes. In addition to the benefit of the increased stability, the server only requires a small amount of volatile memory (an integral commodity on large server systems).

The server contains complex components. As shown in Figure 4, the class Server is the drive class at the server side. The Server creates a ConnectionHandler that waits for a socket connection request from the client and creates one ServerConnection thread for each socket. One ServerConnection instance is responsible for a connection between the server and one client. The

incoming message box is modified. Through the SQLInterface, the ServerDriver accesses the SQL server database.

The client side (Figure 5) has a main class Client. This class creates a WorkspaceWindow at the beginning when a user logs in. In the WorkspaceWindow, a RoleList and a RoleEditor can be opened for chatting with a specific role and the AgentList and the AgentEditor are opened for special roles to manage. The class ClientConnection provides the connection between the server and the client. The WorkspaceWindow contains a MessageList that includes many MessageWindows. Any changes to the state of the system are also updated immediately on the client systems in real time such that it is not necessary for the user to reload any interface windows. This means that the server is not completely passive but actively sending updates to the clients.

B. User Interfaces

The user interfaces are in window-style. A new user can be added to the system by creating an agent. Normally, *user* is used to express the real person and *agent* is used to express the system entity relevant to a user. A user's permissions are a union of the permissions associated with each role the user is playing. The functionality of the interface presented to the user is governed by the permissions of that user. Some buttons are grayed out for those users without the permissions to access based on the role they are playing.

The main user interface is a small window that contains the users' current credits and buttons to access features such as a role list, mailbox, message window, and dialog list. The role list allows users to view roles. The mailbox is where any incoming messages are displayed. The dialog list displays any dialogs that the user is participating in. When a dialog is selected, a list of the participants in that dialog is presented. If the user is the creator of the dialog, s/he has access to the buttons to add or remove roles to the dialog. Each of these interfaces is dynamically updateable, and any changes to the users'

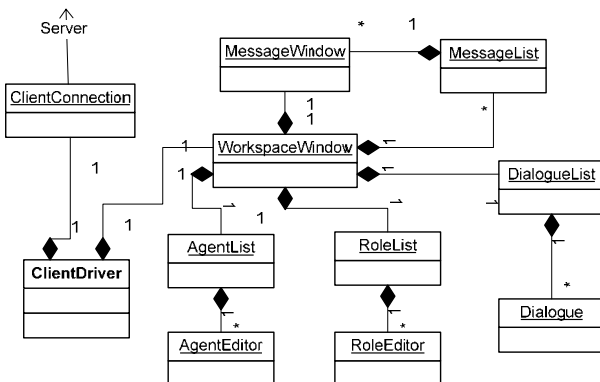


Figure 5. The client side classes.

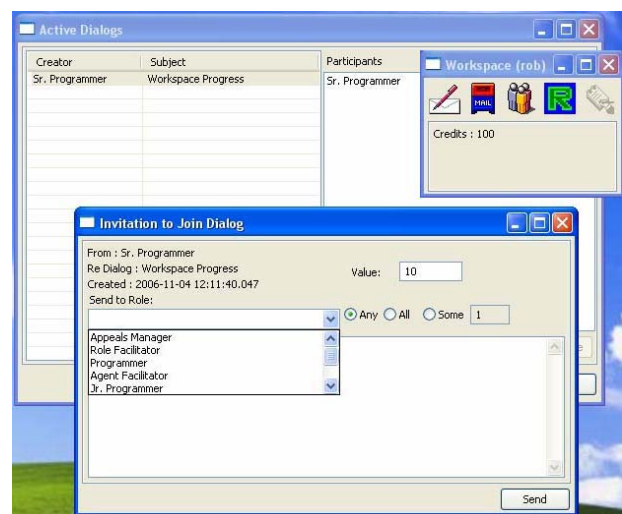


Figure 6. The main user interface.

MessageDispatcher is used to notify the client that its

permissions, credits, or inbox are immediately updated on open windows in time (Figure 6).

When a user composes a new message, a drop down list of the roles s/he is playing is presented. The user chooses the role s/he wishes to play when sending the message. A drop down list of all roles on the system allows the user to select the desired receiving role. The input fields are presented for the number of credits that the sender wishes to award the receiver if an appropriate reply is received, and the number of desired recipients of the message. The user may enter the message text at any time but the message may not be sent until all the required information has been entered.

When the recipient of a message opens a message from their mailbox, all of the information that the sender supplies is displayed but un-modifiable, only a text area for the recipient's reply is available for modification. The message is then marked as a reply and sent back to the sender. Upon receiving a reply, only two options are presented: to either pay, or to deny payment. If "Pay Agent" is selected, a number of credits corresponding to the value of the message are moved from the sender's credits to the receiver's credits.

If the sender of a message receives a reply that the sender deems insufficient, the sender may choose to deny payment, in this case the message is marked "denial of payment" and sent back to the user who initially received the message and composed the reply. The recipient is then presented with two options: to accept the sender's denial of payment, or to appeal the decision of the sender. Appeals are a separate kind of message whose recipient role can be chosen only from those that have the "Appeals Manager" obligation.

When a user creates a dialog (Figure 7), s/he must choose the role that s/he wishes to play within this dialog and assigns a subject to that dialog. Only then can a user compose "Dialog Invitation" messages. Dialog invitations, composed only by the creator of a dialog, consist of the desired recipient role, the number of agents playing those roles whom should receive invitations and the value of the message. A text field is supplied for the dialog creator to outline what is expected of the participant if they are to receive the payment that is specified by the invitation message. Once an agent joins a dialog, they must either be sent a denial of payment or paid the value of their invitation message before they are removed from the dialog or the dialog is closed. The denial of payments is, of course, open to appeal.

The dialog window itself is also dynamically updateable so collaboration may occur in both real time and long term. All entries in the dialog are displayed with the sender role displayed in bold followed by the body of the dialog entry. It is possible for two agents to play the same role within the same dialog, which has been argued as a benefit in Section 2. However, an option is provided to distinguish different agents with the same role by a number assigned when they join the dialog displayed in parenthesis. Any entries made by the user who is viewing the dialog are displayed in red and have the word 'Me' displayed in parenthesis. System messages such as an agent joining a dialog are displayed in blue.

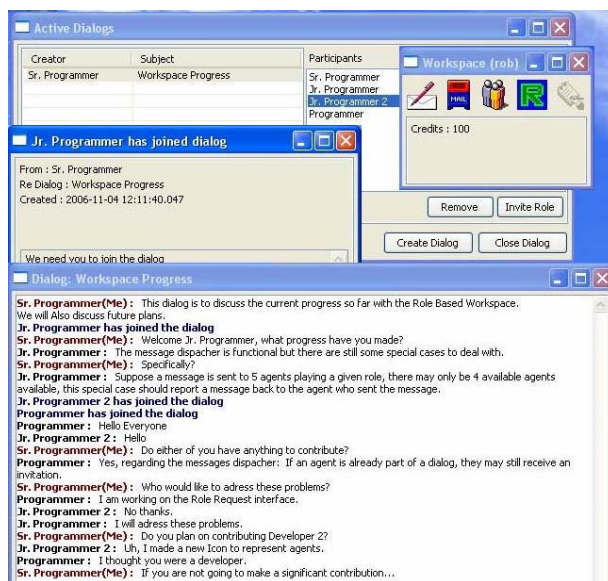


Figure 7. The dialogue interface.

C. Client Interactions

Both the client and server programs are written in Java using the Standard Widget Toolkit (SWT) widgets.

For each client connection, there is a dedicated thread on both the server and the client. Both threads have an outbox queue where network messages from other threads are stored until the communications thread is done dealing with any incoming network messages. This approach avoids any concurrency issues with different threads attempting to communicate simultaneously. Any network messages that require a reply are numbered with a distinct long integer such that message replies can be paired with the appropriate request.

Network messages are passed as strings that are terminated by an escape character, thus message strings may include a new line. Each network message begins with a message header that determines the context of the rest of the message. For example, a login message from the client begins with "LOGIN" header that is immediately followed by a username, password and request number. The client thread that initiates the login then waits for a reply with the same request number. If the login is successful a "LOGIN_CONFIRMED" message consisting of the request number, an agent object and a permissions object is sent from the server. The login initiating thread is then notified that it may resume execution.

Classes such as Agent, Dialog, Message, Permissions, Responsibilities, Role and ListResource (general purpose collection of strings) implement an interface called Resource.

```
public interface Resource {
    public void receive(NetworkReader in);
    public String[] toStrings();
    public String getType();
}
```

This requires all objects of the above classes to implement the receive(...), toStrings() and getType() methods. In this manner such objects may be passed freely over the network. Some of the above classes may

contain other classes of objects, i.e., a role instance contains a Permissions instance and a Responsibilities instance.

The functionality of the system is primarily driven by the client application, objects are maintained on the server by means of REQUEST_RESOURCE, APPEND_RESOURCE and DELETE_RESOURCE network messages that correspond to request, delete, and append properties of the Permissions object. Although this method of access control was intended to be the only access control during early development it was noted that there was a need for many special cases that have to be treated in a less general manner. The above messages apply to objects that are global to the system i.e. the collection of roles and agents or dialog headers.

A dialog object in the system consists of: an identification string, entry number, an agent string, a role string, a date and time string, and the dialog text string. In an attempt to treat dialogs in a general manner, a dialog with an entry number 0 is treated as a dialog header. As more functionality was built into the dialog mechanisms, more data was needed in the dialog header. This caused a conflict of interest between adding more data fields (to both the dialog objects and the database representation of them) and the wasted space that would be contained in dialog objects that were not a header. As the system matured, it became clear that separate object types should have been created. Currently, a dialog header has: the dialog subject in the role field, the agent field contains the dialog creator and the data field contains a list of agents and the role they are playing in that dialog on alternating lines.

The real time updating of the client window is accomplished by a network message that is initiated at server side: the "RESOURCE_CHANGE" header is followed by the type of resource that has changed and the index of the resource. Any client window that has dynamically updateable content implements an interface called updateable, requiring the class to implement an update method. It is through this interface that the client connection thread may execute synchronized code within

the Graphical User Interface (GUI) thread by means of the SWT Shell.asyncExec(...) method.

D. Server Management

To facilitate role-based chatting, management is an important job. Agent management manages all the uses profiles and relevant roles the relevant users are currently playing and have played in the past. Role management deals with all the roles in the system.

The main administration interface is a window that is same as a user interface but with more features such as role and agent management. The role list allows administrators to manage roles. The agent list allows the same actions to be performed on agents. The basic administrative operations for role management are as follows (Figure 8):

- Add/view/edit/delete a role or an agent;
- Approve/reject an application for role;
- Assign /delete a role to/from an agent;
- Adjust the credits of a person; and
- Dispatch messages to people.

A new agent is created with a unique username, password, number of credits, and an empty list of roles. There are default roles for a new agent. Roles can then be added to the agent. With each role that is added to an agent, the user's permissions and obligations grow accordingly.

Role and agent management is implemented through client connections. If a particular user has a role with associated permissions such as add/delete agents, his/her agent may make changes to the agent table in the database. This allows for a connected client to change an agent's password, credits or add/remove roles to that agent. Likewise, if a user has a role that possesses the permission to append/delete roles, his/her agent may change the permissions or obligations for that role.

In summary, this tool supports long term anonymous collaboration in a dynamic, event-driven environment. Features include the encouragement of participants' participation with credits for contribution [9, 17], denial

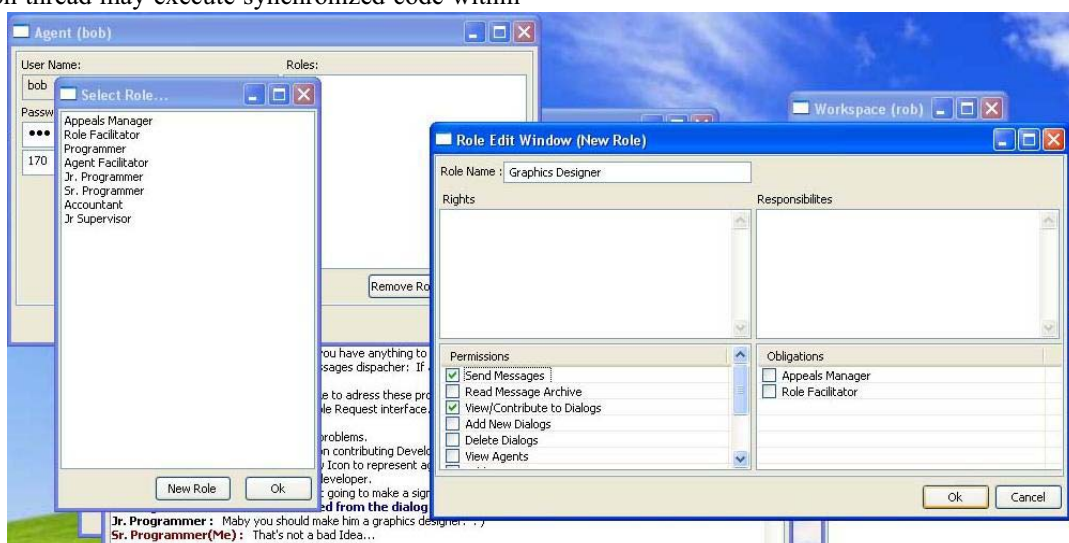


Figure 8. The role management interface.

of payment for insufficient contribution and an appeal mechanism to prevent cheating. The emphasis is on two collaboration mechanisms: messages and dialogs. A message is a one-shot collaboration that involves only two agents with a request and a reply. A dialog is an on-going collaboration that includes many agents.

V. RELATED WORK

Even though roles are good mechanisms to facilitate collaboration, there is little research and practice on introducing roles into chatting tools. Most of the research applies roles in CSCW systems.

In EIES (Electronic Information Exchange System) [16], roles are built out of a subset of the primitive privileges (such as *append*, *link*, *assign* and *use*) that are crucial to a human communication process.

In Quilt [10], roles are introduced in the form of predefined writers (who are allowed to change their own work only), readers (who are not allowed to modify the document), and commentators (who can only add "margin notes" to it).

Patterson [12] emphasizes a role concept with the idea of an interface between objects. Given the roles of users, the messages understood by them are known. In [12], the role is used to enable and disable an object's visibility and to act as a filter on the input events.

Edwards introduces access control policies and roles to avoid chaos in collaborative applications [7]. A role is described as a category of users within the user population of a given application; and all users of a certain role inherit a set of access control rights to objects within the application. A role's dynamic property can change by mapping a role's name to a policy.

Smith et al. built the Kansas system in 1998 and emphasized the importance of roles [14]. People in a group play various roles even though they may not be well defined. They intend to support roles by special treatments in multi-user interfaces. Roles are in general supported by a system's treatment of output and user inputs, which are similar to the view of incoming messages and outgoing messages at a more abstract level.

Looi [11] points out that on-line chatting could address the problem of supporting different roles in a collaboration group. In on-line chatting, various forms of collaboration modes delineate specific roles for each individual. Various ways can be explored to support an individual in enacting or playing her own specific role.

VI. CONCLUSIONS AND FUTURE WORK

Role-based chatting promises many new characteristics improving conventional online chatting.

- Flexible: the users can choose the roles to play according to preferences and the administrator can modify roles based on the changes of the users.
- Anonymous: the users do not have to present their identities. This characteristic helps shy people present their real ideas.
- Manageable: unlike the anonymous posting in the BBS style, the users are manageable by a group

leader. This guarantees that the presentations of anonymous users are convincing and valuable.

- Encouraging: based on special credit setting and computing strategies, people who want to contribute more in the community would like to play more roles and present more to obtain more credits.
- Concentrating: users can concentrate on a special problem without interruptions.

Role-based chatting can be applied in many applications such as client services, virtual communities, collaborative decision making, e-learning and e-training.

As this is a prototype, more work left is to be investigated:

- More powerful role engine is needed to be provided. The current state is to provide credits for agents and help role assignments. Future work includes more regulations for role assignment and more dynamics related with roles and agents.
- System security requires additional attention. Appropriate permissions, violations and messages are sent from the server to the client. This information is sensitive if anonymity of the system is preserved and a client should be restricted to transfer it to a potentially unsecured client.

We also need more comprehensive investigation for the usability and affordance of this tool to evaluate its real value in supporting people's chatting. Before this investigation, we should first make the prototype an online tool that can be easily accessed from the Internet and used in a 365/24/7 style to support collaboration anywhere and anytime.

ACKNOWLEDGMENT

This research is supported in part by National Sciences and Engineering Research Council of Canada (NSERC, No. 262075-06) and the IBM Eclipse Innovation Grant Funding. Thanks also go to Daniel Plourde for participating in the early stage of this work, Mike Brewes for proofreading this paper, and the anonymous reviewers for constructive comments.

REFERENCES

- [1] Bennis, W. and Biederman, P.W., *Organizing Genius: The Secret of Creative Collaboration*, Basic Books, 1997.
- [2] Biddle, B.J., Recent Developments in Role Theory, *Social Annual Reviews*, vol. 12 (1986), pp. 67-92.
- [3] Black, J.S., "Work Role Transitions", *J. of International Business Studies*, vol. 19, no. 2, 1988, pp. 277-294.
- [4] Bostrom, R. P., "Role Conflict and Ambiguity: Critical Variables in the MIS User-Designer Relationship", *Proc. of the 17th Annual Computer Personnel Research Conference*, Miami, Florida, USA, 1980, pp. 88-115.
- [5] Cardwell, J.D., *Social Psychology*, F.A. Davis Co., 1971.
- [6] Cyert, R.M. and MacCrimmon, K.R., Organizations, in *Lindzey, G. and Aronson, E. (Ed.), The Handbook of Social Psychology*, vol. 1, Addison-Wesley, 1968.
- [7] Edwards, W. K., "Policies and Roles in Collaborative Applications", in *Proc. of ACM 1996 Conference on*

- Computer-Supported Cooperative Work (CSCW'96)*, Cambridge, USA, 1996, pp. 11-20.
- [8] Kahn, R. L., Wolfe, D. M., Quinn, R. P., Snoek, J. D., and Rosenthal, R. A., *Organizational Stress: studies in Role Conflict and Ambiguity*, John Wiley & Sons, Inc., New York, 1964.
- [9] Koh, J., Kim, Y.G., Butler, B., and Bock, G.W., "Encouraging Participation in Virtual Communities", *Communications. Of ACM*, vol. 50, no. 2, Feb. 2007, pp. 69-73.
- [10] Leland, M. D. P., Fish R. S. and Kraut, R. E., "Collaborative Document Production Using Quilt", in *Proc. of the Conference on Computer-Supported Cooperative Work*, Portland, OR., USA, 1988, pp. 206-215.
- [11] Looi, C.K., "Exploring the affordances of online chat for learning", *Int. J. Learning Technology*, vol. 1, no. 3, 2005, pp. 322-337.
- [12] Patterson, J. F., "Comparing the Programming Demands of Single-User and Multi-User Application", *The fourth Symposium on User Interface Software and Technology*, ACM Press, Nov. 1991, pp. 87-94.
- [13] Shneiderman, B. and Plaisant, C., "The Future of Graphic User Interfaces: Personal Role Managers", *People and Computers IX, British Computer Society's HCI 94, Glasgow, Scotland, Aug. 1994*, pp. 3-8.
- [14] Smith, R. B., Hixon, R. and Horan, B., "Supporting Flexible Roles in a Shared Space", *The ACM 1998 Conference on Computer-Supported Cooperative Work (CSCW'98)*, Seattle, Washington, USA, 1998, pp. 197-206.
- [15] Turoff, M., Chumer, M., Van de Walle, B. and Yao, X., "The Design of a Dynamic Emergency Response Management Information System", *Journal of Information Technology Theory and Application*, vol. 5, no. 4(2004), pp. 1-35.
- [16] Turoff, M. and Hiltz, S.R., "The Electronic Journal: A progress Report", *Journal of the American Society for Information Science*, Vol. 33, No. 4, July 1982, pp. 195-202.
- [17] Zhu, H. "Encourage Participants' Contributions by Roles", *IEEE International Conference on Systems, Man and Cybernetics*, Oct. 2005, Hawaii, USA, pp. 1574-1579.
- [18] Zhu, H., "Role as Dynamics of Agents in Multi-Agent Systems", *System and Informatics Science Notes*, vol. 1, no. 2, July 2007, pp. 165-171.
- [19] Zhu, H. "Role Mechanisms in Collaborative Systems", *International Journal of Production Research*, vol. 41, no. 1, Jan. 2006, 181-193.
- [20] Zhu, H. and Alkins, R., "A Tool for Role-Based Chatting", *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, Montreal, Canada, Oct. 7-10, 2007, pp. 3795-3800.
- [21] Zhu, H. and Zhou, M.C., "Role-Based Collaboration and its Kernel Mechanisms", *IEEE Trans. on Systems, Man and Cybernetics, Part C*, vol. 36, no. 4, July 2006, pp. 578-589.
- [22] Zhu, H. and Zhou, M.C., "Roles in Information Systems: A Survey", *IEEE Trans. on Systems, Man and Cybernetics, Part C*, vol. 38, no. 3, May 2008, pp. 377-396.
- [23] Zhu, H. and Zhou, M.C., "The Role Transferability in Emergency Management Systems", *Proc. of the 3rd International Conference on Information Systems for Crisis Response and Management*, Newark, NJ, USA, May 14-17, 2006, pp. 487-496.
- Haibin Zhu** is an Associate Professor of the Department of Computer Science and Mathematics, Nipissing University, Canada. He received B.S. degree in computer engineering from Institute of Engineering and Technology, China (1983), and M.S. (1988) and Ph.D. (1997) degrees in computer science from the National University of Defense Technology (NUDT), China. He was a visiting professor and a special lecturer in the College of Computing Sciences, New Jersey Institute of Technology, USA (1999-2002) and a lecturer, an associate professor and a full professor at NUDT (1988-2000). He has published 70+ research papers, four books and one book chapter on object-oriented programming, distributed systems, collaborative systems and computer architecture.
- He is serving and served as co-chair of the technical committee of Distributed Intelligent Systems of IEEE SMC Society, editor for the Int'l J. of Intelligent Control and Systems, member of the Domain Experts Board of International Journal of Patterns, member of the editorial board of International Journal of Software Science and Computational Intelligence, guest editor for the special issue of "Collaboration Support Systems" for IEEE Trans. on SMC(A), guest associate editor for a special issue for the Int'l Journal of Pervasive Computing and Communications, poster co-chair and Organizer of workshop on Role-Based Collaboration (RBC) for the International Symposium on Collaborative Technologies and Systems (CTS 2008), May 19-23, 2008, Irvine, California, USA, vice program co-chair of the IEEE International Conference on Complex Open Distributed Systems (CODS'2007), 22-24 July 2007, Chengdu, China, publicity co-chair, the International Symposium on Collaborative Technologies and Systems (CTS 2007), May 21-25, 2007, Orlando, FL, USA, organizer for the workshop on RBC of the 2006 ACM Int'l Conf. on Computer-Supported Cooperative Work (CSCW'06), organizer of 6 special sessions on RBC for the Int'l Conference on SMC (2003-2008), and program committee member for more than 20 international conferences. His current research interests include role-based collaboration, collaborative systems, software engineering, and distributed intelligent systems.
- Dr. Zhu is a senior member of IEEE, a member of ACM, and a life member of the Chinese Association for Science and Technology, USA. He is the receipt of the 2006-2007 research award from Nipissing University, the 2004 and 2005 IBM Eclipse Innovation Grant Awards, the Best Paper Award from the 11th ISPE Int'l Conf. on Concurrent Engineering (ISPE/CE2004), the Educator's Fellowship of OOPSLA'03, a 2nd Class Nation-Level Award of Education Achievement from Ministry of Education of China (1997), a 2nd Class Nation-Level Award of Excellent Textbook from the Ministry of Education of China (2002), three 1st Class Ministry-level Research Achievement Awards from The Commission of Science Technology and Industry for National Defense of China (1997, 1994, and 1991), and a 2nd Class Excellent Textbook Award of the Ministry of Electronics Industry of China (1996).
- Rob Alkins** is a senior student of Nipissing University.
- Matthew Grenier** is a senior student of Nipissing University.