An Eclipse Plug-in for Role-Based Collaboration

Haibin Zhu
Department of Computer Science and Mathematics, Nipissing University, 100 College Drive, Box 5002, North Bay, ON P1B 8L7 Canada
(705)474-3461 ext 4434
haibinz@nipissingu.ca

Bart Verzijlenberg
Department of Computer Science and Mathematics, Nipissing University, 100 College Drive, Box 5002, North Bay, ON P1B 8L7 Canada
orioniz@gmail.com

ABSTRACT

Role-Based Collaboration (RBC) 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. RBC imposes challenges and benefits not found in traditional CSCW systems. A tool or environment for RBC is required for research and practice of RBC.

In this paper, we present an Eclipse plug-in that implements the E-CARGO model and supports RBC.

Categories and Subject Descriptors

K.4.3 [**Organizational Impacts**]: Computer-supported Collaborative work

General Terms

Management, Design, Experimentation, Human Factors

Keywords

Role, Role-Based Collaboration, RBC, CSCW, Eclipse

1. INTRODUCTION

Role-based collaboration (RBC) has been proposed for a new paradigm of collaboration [1, 2]. RBC has shown many benefits and advantages in different kinds of collaborations among people, system components, computers and agents. To make RBC a reality, it is necessary to provide tools or platforms to support RBC. Our practice on RBC has shown that it is not only possible but also beneficial in improving collaboration with computer systems.

Based on our E-CARGO model, the scenario of RBC is as follows:

1. A computer system manages (stores and retrieves) a lot of roles including primitive roles [2] and user-defined roles.
2. A person signs into the system, gets an account from the system (creates an agent for him/herself), and is assigned a default role by the system.
3. This person selects some of the listed roles and applies for them.
4. The person who is playing the facilitator role approves the applications.
5. ……
6. At some time point, many people are playing many roles.
7. … …
8. A person who is in charge of environments creates environments for future collaborations.
9. A person who wants to organize collaboration creates a group based on an environment and sends messages to the roles relevant to this group. This person is a group leader.
10. A person who wants to collaborate joins a group and sends messages to the roles. This person is called a group member.
11. The system or the person who is playing the dispatcher role dispatch messages to relevant agents (persons).
12. The person (agent) who obtains the messages replies.
13. ....
14. Every person works in a group by
    - sending, receiving and replying messages;
    - creating, deleting and accessing the objects in the group; and
    - leaving the group.
15. The group leader adjusts the collaboration by approving roles, assigning roles, dispatching messages and evaluating the group members (agents).
16. The group leader freezes the group to stop the collaboration.
17. … …

Note that, only the group leader deals with the people. Other group members only work with roles. This is the key feature of RBC. Leaderships are facilitated by managing roles and role assignments to people.
In this position paper, we present our practice to prototype a tool for people to practice RBC. This prototype is approaching to a platform to accommodate RBC.

2. THE COMPONENTS OF THE PLUG-IN

2.1 E-CARGO Package

The E-CARGO model specifies the required components of a role-based collaboration (RBC) system [2]. Since the model is general in nature and may be applied to a variety of applications, it was decided that the model should be implemented separately from the eclipse plug-in. The RBC kernel [2] implements all the required abilities and mechanisms of the E-CARGO model, but does not provide for any specific use of the model nor a user interface. This has the added benefit of allowing changes to be made to the kernel without necessarily affecting the plug-in.

2.2 RMI Server

Due to the distributed nature of the E-CARGO model (multiple agents/roles must be able to interact from various locations), a central server was required to handle all access to the E-CARGO kernel. Remote Method Invocation (RMI) was selected to allow seamless calling of methods by the remote plug-ins. To allow the E-CARGO package to remain generic, it was wrapped to conform to RMI requirements. All functionality is accessible through the RMI interface. The server shows a log of most remote method invocations, as well as the time that they were logged. This allows for the detection of bottlenecks. For example, exporting the main object on the server takes less than a tenth of a second when the code base is accessible through a web server, whether local or remote. However, when a direct link to the hard drive was provided instead, the same actions blocked the server for 9 seconds

2.3 CVS Proxy

One need for the plug-in is the ability to share resource (projects, source codes, documents etc). Concurrent Version System (CVS) is a near perfect candidate for this task, since it stores resources and their change history in an efficient way. In addition it is designed for collaboration among multiple users, and is tightly integrated with Eclipse. However, as mentioned, it is near perfect. CVS allows changes to be committed as long as the user is authenticated. The role the user is playing is not considered in the request. Preventing agents from committing on the client side is prone to being bypassed. In addition, an agent may only have access to a limited subset of the resources available on the CVS server. For that reason a server side solution it required. By using a proxy server instead of accessing the CVS server directly, it is possible to tightly control the agent's actions on the CVS server. When the agent connects to the proxy server, the username and password are used to login to the local RMI server, which then can be used to accept or deny all or part of the request, as well as filtering the response from the server, i.e., Internet Explorer (IE) inaccessible projects/files can be removed from the response. In this way the CVS client does not require any modification and becomes role-based automatically. In addition the choice of CVS server becomes arbitrary, since the role-based control is implemented in the proxy.

2.4 Eclipse Plug-in

The Eclipse plug-in provides a graphical interface to the RMI server as well as several other convenient functions. The plug-in provides two main views: RBC View and RBC Object View. In addition, it also provides a builder and two wizards. The remainder of this report discusses the Eclipse plug-in.

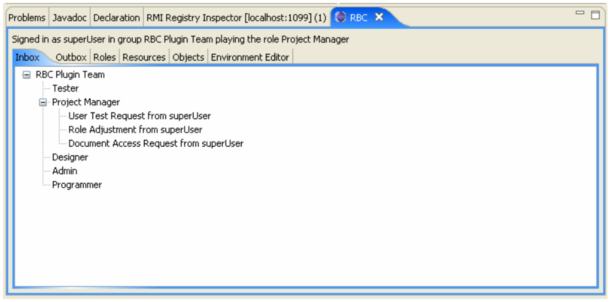
3. THE CLIENT VIEW OF RBC

Once eclipse is connected to the RMI server, the RBC view shows a status line that shows the name of the current agent and the agent's current group and role. Below the status line are the following tab items:

- Inbox: The agent's inbox
- Outbox: The agent's outbox
- Roles: The roles that the agent can apply for
- Resources: The accessible resources for the agent's current role
- Objects: The accessible objects that the agent can use.

3.1 The Inbox and the Outbox

The Inbox (Fig. 1) tab shows each of the groups that the agent is a member of, as well as the playable roles in each of the groups. The role currently being played by the agent is opened and displays the received messages for the role. Selecting a message opens it in an RBC Object View (See 3.4). Right clicking any of the roles displays a context menu, if the role is currently being played by the agent, then a "Create new Message" options is shown. If the role is not currently being played by the agent then an "Assume Role" option is available. No actions can be taken for any role unless it is currently being played by the agent.



Signed in as superUser in group RBC Plugin Team playing the role Project Manager

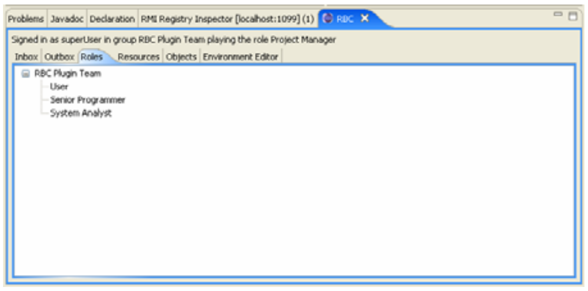
RBC Plugin Team
Tester
Project Manager
User Test Request from superUser
Role Adjustment from superUser
Document Access Request from superUser
Designer
Admin
Programmer

Fig. 1 The Inbox of an Agent

The Outbox tab functions exactly the same as the Inbox, however, it shows the Outbox instead of the inbox. The context menus remain the same.

3.2 Roles

The Roles tab (Fig. 2) shows all the groups in the system, as well as any roles that the agent is not currently able to play. Right clicking a role shows a context menu with the option of sending a role application request to the administrator of the group.

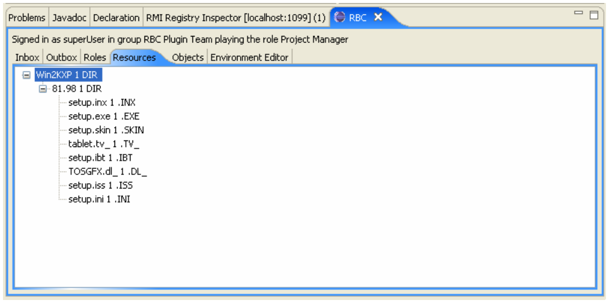


Signed in as superUser in group RBC Plugin Team playing the role Project Manager

RBC Plugin Team
User
Senior Programmer
System Analyst

Fig. 2 Roles

3.3 Resources



Signed in as superUser in group RBC Plugin Team playing the role Project Manager

Win2KXP 1 DIR
81.98 1 DIR
setup.inx 1 .INX
setup.exe 1 .EXE
setup.skin 1 .SKIN
tablet.tv_ 1 .TV_
setup.ibt 1 .IBT
TOSGFX.dl_ 1 .DL_
setup.iss 1 .ISS
setup.ini 1 .INI

Fig. 3 Resources

The Resources tab (Fig. 3) shows the accessible resources for the agent's current role. A tree structure is shown that mirrors the file structure on the hard drive. Any file that is not accessible in any way to the role is not shown. In this way the agent is not bothered with resources that are only accessible to other roles. Multiple projects can exist in the same location in this way without being visible to agents not in the group.

3.4 Objects

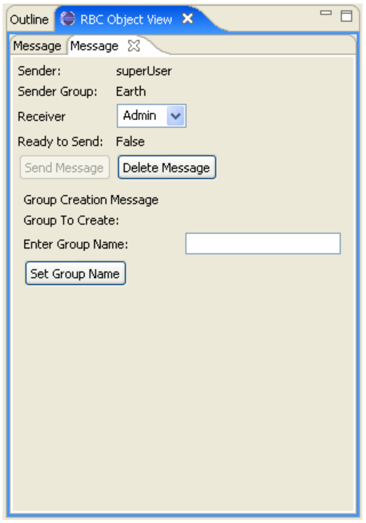
The Objects tab (Fig. 3) shows the objects accessible to the current role. RBC objects are running instances of classes on the server side that provides the functionality to the agent. They are also an easily extendible interface to the RMI interface. The plug-in provides wizards for the creation of new objects. By clicking the Objects tab, an RBC Object View (Fig. 4) is popped up.

The RBC Object View

The RBC Object View (Fig. 4) provides a place for objects and messages a location to draw their interface. Each tab contains a viewer for a single object or message. If the tab is closed the viewer is disposed and not re initialized until the object/message is viewed again. Each object/message viewer is provided with a Standard Widget Toolkit (SWT) Composite to draw in.

Messages have a two line banner at the top indicating who sent the message and what group the message was sent from.

If the message has not been sent yet then a receiver selector box, a ready to send status line and two buttons “Send Message” and “Delete Message” are also shown.



Sender: superUser
Sender Group: Earth
Receiver: Admin
Ready to Send: False

Group Creation Message
Group To Create:
Enter Group Name:

Fig. 4 Objects

As mentioned, RBC objects are running on the server. This allows them to interact directly with the server and the E-CARGO package. Depending on the agent’s current role the agent can make changes to the system as required. Objects have several components, namely the main object that is instantiated on the server side, an interface for the client to access the object and a viewer to view the object on the client side.

The class files for objects are stored on the server and are not shipped with the plug-in code. Instead they are downloaded dynamically from the server as required. In this way the plug-in always runs the latest code from the server. It is also possible to swap in new object class definitions while the client plug-in is active. This allows for easy updating of code without requiring either the server or the client to stop.

As mentioned, in order to display the objects to the agent a client side viewer is required. The client side viewer communicates with its server side counterpart. Due to the fact that the viewer is not running in the display thread of the client, it is required to run any display code as an asynchronous run-able in the display thread. This has the advantage of allowing the server object to offload CPU intensive tasks to the client without slowing down the user’s interface.

Messages

Much the same way as dealing with objects, messages are always stored and running on the server side. To allow the client to view and modify the message a client side viewer is required. The viewer makes the required changes to the message before instructing the server to deliver the message to the receiving role. When the message is received by the receiver, the viewer is again instantiated. The viewer may show different views depending on whether the message is being created or received.

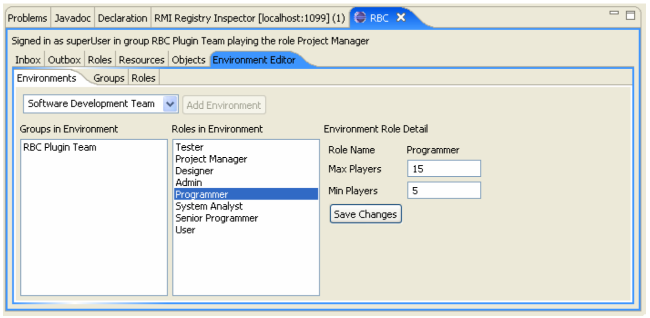
Wizards and New Code

To simplify the addition of RBC objects and new message types, the plug-in comes with a basic wizard for each that creates a minimal object/message and adds it to the Eclipse workspace. When the object/message is ready to be used in the system a function is provided (as long as the current role is allowed to add new code to the server) to upload the class files to the server and initialize the object/message on the server. The object/message is then accessible to any plug-in for dynamic download, provided the role active on the client has access to the object/message. The new code is also forwarded to any client running an old version of the code. The new version is swapped in as soon as the viewer of the object/message is closed.

3.5 Environment Editor

The Environment Editor tab (Fig. 5) is shown to the user only if the user is authorized to make changes to the environment. The environment editor is used to add new environments, groups and roles, as well as change any existing ones.

Environments Tab



Signed in as superUser in group RBC Plugin Team playing the role Project Manager

Software Development Team

Groups in Environment
RBC Plugin Team

Roles in Environment
Tester
Project Manager
Designer
Admin
Programmer
System Analyst
Senior Programmer
User

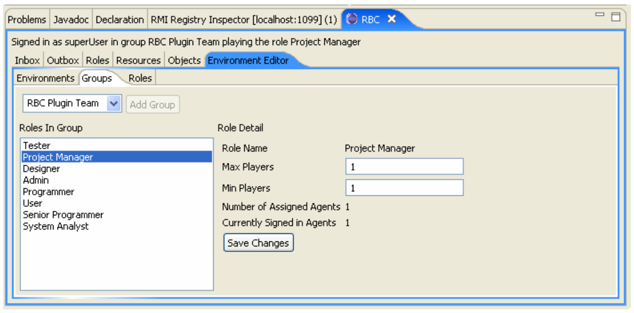
Environment Role Detail
Role Name: Programmer
Max Players: 15
Min Players: 5

Fig. 5 Environment Management

The Environments tab (Fig. 5) of the environment editor allows authorized agents to create new environments and modify existing environments. Environments define a subset of all the roles in the system. This subset of roles is then used to define the base for a group. Groups based on the environment can only use roles in the subset. Each of the roles included in the environment define a minimum and maximum number of players. These values can be changed in the Environments tab. In addition, roles can be added to the environment from this tab.

Groups Tab

The Groups tab (Fig. 6) displays all the groups in the system, and the roles they contain. Each of the groups is based on one of the environments defined in the system. A group contains a subset of the roles defined in its base environment. Each of the roles also has a minimum and maximum number of players defined, separately from its base environment. If a group's base environment is modified, then the changes are reflected in the group. However, changes made in a group are not reflected in the group's base environment.



Problems | Javadoc | Declaration | RMI Registry Inspector [localhost:1099] (1) | RBC

Signed in as superUser in group RBC Plugin Team playing the role Project Manager

Inbox | Outbox | Roles | Resources | Objects | Environment Editor

Environments | Groups | Roles

RBC Plugin Team | Add Group

Roles In Group
- Tester
- Project Manager
- Designer
- Admin
- Programmer
- User
- Senior Programmer
- System Analyst

Role Detail

Role Name	Project Manager
Max Players	1
Min Players	1
Number of Assigned Agents	1
Currently Signed in Agents	1

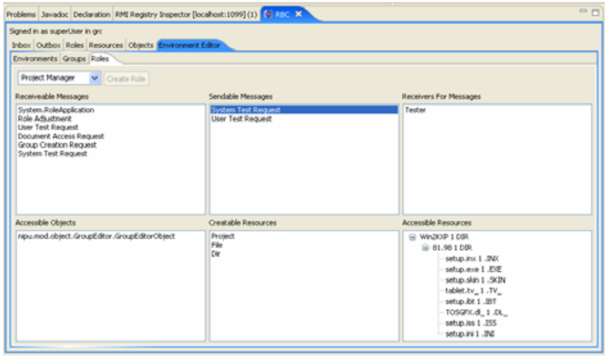
Save Changes

Fig.6 Groups Management

The Groups tab allows the creation of new groups. When a new group is created, the user must specify what base environment to use for the group. In addition, the user may add or remove roles from the group. Again, only roles defined in the base environment can be added to the group.

Since agents play roles in groups, each of the roles also tracks the number of agents that are able to play the role, and the number of agents who are currently playing the role.

Role Management



Problems | Javadoc | Declaration | RMI Registry Inspector [localhost:1099] (1) | RBC

Signed in as superUser in grc

Inbox | Outbox | Roles | Resources | Objects | Environment Editor

Environments | Groups | Roles

Project Manager | Create Role

Receiveable Messages
- System.RoleApplication
- Role Adjustment
- User Test Request
- Document Access Request
- Group Creation Request
- System Test Request

Sendable Messages
- System Test Request
- User Test Request

Receivers For Messages
- Tester

Accessible Objects
- nipu.mod.object.GroupEditor.GroupEditorObject

Creatable Resources
- Project
- File
- Dir

Accessible Resources
```
Win2KXP 1 DIR
  81.98 1 DIR
    setup.inx 1 .INX
    setup.exe 1 .EXE
    setup.skin 1 .SKIN
    tablet.tv_ 1 .TV_
    setup.ibt 1 .IBT
    TOSGFX.dl_ 1 .DL_
    setup.iss 1 .ISS
    setup.ini 1 .INI
```

Fig. 7 Role Management

The Roles tab (Fig. 7) is used to actually define all the roles in the system. It is therefore the most involved of the tabs, and most of the environment editing tasks are done from this tab. At the top of the tab is a role selection box. When a role is selected in the box, the role's information is loaded into 6 property boxes. They are the role's receivable messages, send-able messages, receivers for messages, accessible objects, creatable resources, and accessible resources. Each property box provides appropriate context sensitive popup menus to make desired changes.

The role's receivable messages are all the in messages that this role is able to receive. In the same way, the role's send-able messages are all the messages that the role is able to send. However, simply being able to send the message is not enough. The role must also have someone to send the message to. That is where the receiver property box comes in. A role must have permission to send a message to a particular role. In order for a message to be added to a role's receiver list, the message type must first be added to a role's receivable messages. In this way, a programmer for example would not be able to send a "System Test Request" message to a tester. Instead, the programmer can only send the message to the project manager. Even though both the tester and the project manager are capable of receiving the message, the programmer does not have the required authority to send the message to the tester.

In order for a role to be able to access an object on the server, the role must first be given permission to access those objects. The "Accessible Objects" box provides the ability to add those permissions.

The creatable resources box shows all the resource types that a role is allowed to create. This functionality interacts with the CVS proxy server, mainly when committing new resources. The CVS proxy server can check with the RMI server whether or not the role is allowed to add the resource type. For example, only the project manager should be allowed to create a new project on the server. If the CVS proxy is asked to create a new project on the server, and the role is not authorized to do so, then the request fails and an appropriate error message is displayed.

The accessible resources box shows all the resources in the system, as well as their type and the access permission for the resource. Currently the only access types are "0: No Access" and "Not 0: Access" allowed. If a role is not allowed to access a resource it is not shown to the agent. The accessible resources box allows the access permissions to be changed for any resource. If a role is granted access to a resource, but does not have access to the parent resource, then the role is automatically granted access to the parent resource.

3.6 Storing System Settings

```
<Role name="System Analyst">
    <AccessibleResources name="root" />
    <OutMessage name="System.RoleApplication">
        <Receiver name="Project Manager" />
    </OutMessage>
    <OutMessage name="Document Access Request">
        <Receiver name="Senior Programmer" />
    </OutMessage>
    <OutMessage name="Role Adjustment" />
</Role>
<Role name="Senior Programmer">
    <CreatableResource name="File" />
    <CreatableResource name="Dir" />
    <AccessibleResources name="root">
        <Resource name="/Win2KXP" permission="1">
            <Resource name="/Win2KXP/81.98" permission="1">
                <Resource name="/Win2KXP/81.98/setup.inx"
                    permission="1" />
                <Resource name="/Win2KXP/81.98/setup.exe"
                    permission="1" />
                <Resource name="/Win2KXP/81.98/setup.skin"
```

Fig. 8 The XML File

When the server is started it reads an Extensible Markup Language (XML) file (Fig. 8) containing all the environment, group and role settings and configures the system to be the same as when the server was shutdown last time. As the small portion of the XML file above shows, the settings are stored in plain text, and are logically ordered. This allows a human system administrator to adjust or remote settings outside of the eclipse plug-in. Since XML is a popular format, and is easily accessible it was a natural choice to use for storing system settings. The format uses a hierarchical organization of information, as well as tags to identify all stored information, making it easy to identify a chunk of related settings.

4. CONCLUSIONS AND FUTURE WORK

Role-based collaboration (RBC) is an emerging inter-discipline research and practice topic that will give rise to significant innovations to such areas as CSCW, programming languages, and collaborative intelligent systems. The above mentioned Eclipse Plug-in makes RBC possible. Clearly there is more work left to be done. The following lists some of the work that can/needs to be done:

- CVS Proxy server support: At this time the proxy server is able to correctly identify the commands being sent by the client and communicate with the RMI server. Future work needs to expand this by retrieving the appropriate access information from the RMI server and then allowing, filtering or blocking each of the possible commands. This would allow any CVS server to be converted immediately to a RBC CVS Server.
- Develop objects and messages to be used with the plug-in: As of the time of this writing, the E-CARGO package, RMI server and Eclipse plug-in have been written to be as generic as possible. This allows the project to be expanded easily, and adapted to specific needs. More specific objects and messages are required to support those specific needs.
- Improve the message dispatcher: The current message dispatcher is basic. The dispatcher should be changed so that messages are routed to an appropriate agent playing the role, not just any available agent.

5. ACKNOWLEDGMENTS

This research is partly supported by Natural Science and Engineering Research Council of Canada (NSERC, Grant No. 262075-06) and the IBM Eclipse Innovation Grant Funding.

6. REFERENCES

[1] Zhu, H., Role Mechanisms in Collaborative Systems, *International Journal of Production Research*, 44, 1 (2006), 181-193.

[2] Zhu, H. and Zhou, M.C., Role-Based Collaboration and its Kernel Mechanisms, *IEEE Trans. on Systems, Man and Cybernetics, Part C*, 36, 4(July 2006), 578-589.

[3] Zhu, H. and Zhou, M.C., Supporting Software Development with Roles, IEEE Trans. on Systems, Man and Cybernetics, Part A, to appear (2006).