ROLE-BASED COLLABORATION

Sponsors:
NSERC (National Sciences and Engineering Research Council, Canada)
IBM Eclipse Innovation Grant

Haibin Zhu, PhD
Associate Professor, Senior Member, IEEE
Dept. of Computer Science and Mathematics, Nipissing University, 100 College Dr., North Bay, ON P1B 8L7, Canada
haibinz@nipissingu.ca
http://www.nipissingu.ca/faculty/haibinz

Contents
- The argument of role-based collaboration (RBC)
- The role concepts
- Why do we propose RBC?
- What do we mean by RBC?
- What are the RBC principles?
- How can we support RBC?
- The realm of RBC
- Role-based multi-agent systems
- Role-based software development
- The state of arts and future of RBC
Arguments against on RBC

- A completely negative comment is that role-based collaboration is meaningless because collaboration itself implicates role assignments and role specifications.
- Some others think that roles have been introduced into information systems for more than twenty years and all the problems have been solved.
- Even others state that the software with roles is considered as naziware that is not welcome.
- Even others argue that roles are not encoded solely in human biology or in physical law, roles are devils, and it is almost impossible to describe what roles are.

The Role Concepts
Quotations from Confucius

- “名不正，则言不顺；言不顺，则事不成。”
- “If terminology is not corrected, then what is said cannot be followed. If what is said cannot be followed, then work cannot be accomplished.”
- ----Confucius, 205 BC, China

The role theory from Confucius

- “君君、臣臣、父父、子子”
- “Let the ruler be a ruler, the minister be a minister, the father be a father, and the son be a son”
- [Lun Yu: Yan Hui No. 12, Section 11].
Quotations from W. Shakespeare

- All the world’s a stage,
- And all the men and women merely players;
- They all have their exits and entrances;
- And one man in his time plays many parts.

-----As You Like It, Act II, Scene 7

What are roles?

- “The part or character one has to play, undertakes, or assumes”;
- “The part played by a person in society or life”; or
- “The typical or characteristic function performed by someone or something”.
- “The behavior that an individual feels it appropriate to assume in adapting to any form of social interaction; the behavior considered appropriate to the interaction demanded by a particular kind of work or social position.”

---Oxford English Dictionary
The properties of roles

- A role is independent of objects. We can define it separately. It is a common idea that a role is dependent of objects in object systems. In collaboration, however, collaborators may not care about a specific person. They only want to contact a person who plays a specific role.
- A role should consider both responsibilities (the service interface) when the human player is taken as a server and rights (the request interface) when the human player is taken as a client. That is to say, to specify a role, we must specify both aspects.
- A role can be performed by one or many players at the same time.
- A role can be created, changed and deleted.

The attributes of roles

- Rights: Roles are entities that facilitate human users (principles, subjects) to access system resources (files, objects, and devices).
  - Applied in RBAC or System Management
- Responsibilities: Roles are entities that express different aspects of an object at different contexts at different time point. They provide different services to the outside worlds.
  - Applied in Object or Agent Modeling
- Both rights and responsibilities: in social psychology, people live in a society should take responsibilities and hold rights when playing a role.
  - Applied in Social Psychology
**Interfaces or processes**

- Interfaces: Roles are entities to express the interfaces between objects or agent in collaboration among objects and agents. In this sense, roles only specify what the services are and what the requests are. How the services and requests are processed depends on the role players.
  - Applied in describing object/agent collaboration
- Processes: Roles are behavior describers in specify object or agent abilities. At this point, roles specify not only what services and requests are but also how services and requests are processed.
  - Applied in process specification

---

**The basic viewpoint on roles**

![Diagram of incoming and outgoing messages through an interface with a human in collaboration]

- Incoming Messages
- Service Interface
- Request Interface
- Outgoing Messages
- A Human in Collaboration
The occurrence of roles in a system

A collaborative system including objects, agents, groups and roles

Categories of roles

- Modeling Roles
- Agent Roles
- RBAC Roles
- CSCW Roles
Modeling roles

- Roles are a concept or specialization of a concept.
- Roles can be acquired and abandoned independently of each other.
- Roles can be organized in hierarchies, generalized or specialized.
- Roles model a perspective on a phenomenon.
- Roles are bound to an existing object.
- Roles are used to emphasize how entities interact with each other.
- Roles can be dynamic, involving sequencing, evolution, and role transfer.
- An object may play several roles at a time.
- Various roles of an object may share common structure and behavior.
- The states and features of an object can be role-specific.

Agent roles

- A role can be taken as an interface.
- A role can be taken as a process.
- A role instance is deleted when an agent is destroyed, i.e., its lifetime depends on its agents.
- Roles are used to form different interfaces for agents in order to restrict the visibility of features and to handle permissions for the access to the internal state and role services of agents.
- Roles have three functions: comprise special behavior, form the behavior of an agent, and take a position in a group of agents.
- A role specifies a position and a set of responsibilities which are made up of services and tasks.
- Roles can be used for expressing the organizational structure of a multi-agent system.
- Roles can be used for specifying interactions in a generic way.
- Roles can be used as agent-building blocks in class diagrams.
RBAC roles

- Least Privilege: It requires that users be given no more privileges than necessary to perform their job function.
- Separation of concerns: (1) a role can be associated with an operation of a business function only if the role is an authorized role for the subject and the role was not reassigned previously to all of the other operations; (2) a user is authorized as a member of a role only if that role is not mutually exclusive with any of the other roles for which the user already possesses membership; and (3) a subject can become active in a new role only if the proposed role is not mutually exclusive with any of the roles in which the subject is currently active.
- Cardinality: The capacity of a role cannot be exceeded by an additional role member.
- Dependency constraints: There is a hierarchy or relationships among roles such as contains, excludes and transfers.

CSCW roles

- In CSCW there is neither commonly-accepted concept of roles nor methods to express a role.
- Some use roles as a commonly understandable word;
- Some support role ideas with some special user interface setting mechanisms;
- Some propose a tool to specify roles dynamically.
- Whatever kinds of roles are considered, all the applications use roles to support the human-computer and human-human interactions. Therefore, roles are considered as interaction media in collaborative systems.
Roles as a computational thinking concept

- a fundamental modeling concept;
- reflect the evolution of objects; express the interfaces among objects; and
- facilitate the separation of concerns.

Why Do We Propose RBC?
CSCW systems

- Similar terminologies:
  - CSCW (Computer-Supported Cooperative Work)
  - GSS (Group Support System) or GDSS (Group Decision Support System)
  - CMC (Computer Mediated Communication)
  - Groupware
  - VR (Virtual Reality) or CVE (Collaborative Virtual Environment)

CSCW systems (cnt’d)

- Synchronous systems:
  - Same time, different places
  - Simulate FTF meetings, classrooms, ...
  - MCAS, MSN, NetMeeting, ...

- Asynchronous systems:
  - Any time, any places
  - Simulate letters, bulletin board, ...
  - BBS, WebBoard, WebCT, ...
Goals of CSCW systems

- To support collaboration, we need special methods, tools and techniques.
- CSCW systems should
  - not only provide virtual face-to-face collaboration environment among people at a distance
  - but also improve face-to-face collaboration by providing more mechanisms to overcome the drawbacks of face-to-face collaboration among people.

Problems in current CSCW systems

- Synchronous:
  - Not satisfactory in real application
    - unsatisfactory communication
    - frustrated waiting
    - uneasy environments for discussions
    - complex operations.
    - clumsy, not practical, and frustrating compared to face-to-face collaboration
  - Few human factors considered
- Asynchronous:
  - Few consistent role concepts
  - Few practical tools to support roles management and collaboration based on roles
The problems to apply roles in FTF (face-to-face) collaboration

- Role ambiguity
  - Role ambiguity describes a situation in which the desired expectations sent to the focal person were vague, ambiguous, and/or unclear, thereby making it difficult for the person to fulfill the requirements.

- Role conflict
  - Ideally, consensus and clarity would exist among the expectations of the interested parties. In reality, such a situation is rarely achieved and some conflict between expectations and ambiguity about role requirements is typical.

Requirements from Collaborative Agents

- Multi-agent systems are complex ones.
- Collaboration is a complex task.
- Roles are a way to simplify complex tasks and providing ways to design complex collaborative systems.
What Do We Mean By RBC?

What is RBC?

- Role-based collaboration (RBC) is a computational thinking methodology that mainly uses roles as underlying mechanisms to facilitate abstraction, classification, separation of concern, dynamics, and interactions. It will find wide applications in different fields, such as, organizations, management systems, systems engineering, and industrial engineering.
- RBC will be a fundamental thinking component similar to recursion, abstraction, decomposition, tracking, prefetching, caching, and silence of failure.
- RBC is a way to understand the complexity in Natural, Built, and Social Systems.
- RBC is a fundamental methodology to build virtual organization.
- RBC is a way to extract data to knowledge.
Roles are tools in cognitive activities

- People understand and recognize others by the roles they are playing and have played.
  - Because roles have common senses, but people are unique.
  - By checking the roles others played and playing, we can know the basic properties, backgrounds, and personalities.
  - That is why people would like to list titles in their curriculum vita or resumes.
- We can call this as role-based recognition.

The basic idea of role-based collaboration (RBC)

- If users can
  - clearly know what objects they can access with specific rights
  - can also know which users they can manage or communicate with
- They can then accomplish their jobs meaningfully and efficiently.
- Note: this is similar to problem solving. To solve a problem, you need to know what and where the problem is at first, then know how to solve it and finally solve it.
The procedure of RBC in our society

- **Step 1**: negotiate roles. People discuss or negotiate to specify the roles relevant to collaboration. If a compromise or an agreement is obtained then the collaboration continues to step 2 else it aborts.
- **Step 2**: assign roles. Every person is assigned one or more roles. If agreement is obtained then the collaboration continues to step 3 else it aborts.
- **Step 3**: play roles. People work according to their roles until collaboration completes successfully or some conflicts or discontents occur.
  - **Step 3.1**: check incoming messages. People understand what to respond to the incoming messages confined by the service interface. If conflicts or discontents occur, goes to step 1.
  - **Step 3.2**: issuing outgoing messages. To provide services, people need to interact with the environment by sending messages. If there are no incoming messages, the people could think and issue messages confined by the request interface. If conflicts or discontents occur, goes to step 1.

Case 1: a company

- **Step 1**: Before entering the company, negotiate roles. The person and the company negotiate the roles of the person in the company. If they get an agreement, the company recruits the person and assigns the roles and the collaboration continues to step 2 else it stops.
- **Step 2**: In a company, play the roles until the natural end comes (the person retires or the company is closed) or discontents occur. If the company or the person finds something unpleasant, they negotiate the roles and adjust the roles or transfer the roles. If there is a compromise, the collaboration continues to step 2 else it goes to step 3.
- **Step 3**: The person resigns from the company or the company fires the person.
Case 2: a meeting

- **Step 1:** before the meeting, negotiate roles. The people must negotiate or understand the roles in the meeting. If there is an agreement, the roles are assigned, a meeting is scheduled, and the collaboration continues to step 2 else it stops.

- **Step 2:** in the meeting, play the roles. Each person plays specific roles until the normal end comes (time is out or all the problems are resolved). If there are some conflicts, the roles are negotiated and the roles are reassigned. If there is a compromise, the collaboration goes to step 2 else it goes to step 3.

- **Step 3:** exceptions in the meeting. The person leaves the meeting or the meeting dismisses the person and the collaboration continues to step 2. The meeting might be adjourned and the collaboration stops. Note: The participants may need to negotiate their roles in order to make the next meeting more successful.

The properties of RBC

- **Clear and strict role specification:** it is easy for human users to understand their responsibilities and rights.

- **Flexible role transition:** it is flexible and easy for a human user to transfer from one role to another role.

- **Flexible role facilitation:** it is easy for role facilitators to specify roles. Because a system is developing, even the existing roles might be required to adjust to correspond with the development of the system.

- **Flexible role negotiation:** it is easy to negotiate the specification between a human user and a role facilitator.
Object principles

- O1: Everything in the world is an object. An object can be used to express everything in a collaborative system.
- O2: Every system is composed of objects and a system is also an object.
- O3: The evolution and development of a system is caused by the interactions among the objects inside or outside the system.
- O4: Objects can be created, modified, and deleted.
- O5: A message is a way to activate services of an object.
- O6: An interface is a list of message patterns.
- O7: The interactions among objects are expressed by sending messages that are requests to invoke objects’ actions.
- O8: Each object is an instance of a class which shows the commonality of a group of objects.
- O9: A class X may inherit another class Y. Y is called a superclass while X is called a subclass.
- O10: Classes can be taken as templates of objects.
Agent principles

- A1: An agent is a special object that represents the existence of a human user in RBC.
- A2: An agent can be created, modified, and deleted.
- A3: An agent is autonomous. It can help a human user play some roles based on the collaboration situation.
- A4: An agent is an assistant to a human user. It can accept the human user's command and play future roles such as "play role x at time t".
- A5: An agent is a history record for its corresponding human user in collaboration by storing the roles played.
- A6: RBC may still advance even if some human users log out (not on-line) because their agents can do some jobs on their behalf.
- A7: Agents are flexible. Not all actions of agents are predicted. They can dynamically choose which actions to invoke, and in what sequence, in response to the state of its environment (Jennings et al., 1998; Russell and Norvig, 2003).
- A8: Agents are mobile. They are able to transport them from one site to another in a system (Etzioni and Weld, 1995).

Role principles

- R1: A role is independent of agents.
- R2: A role can be created, changed and deleted.
- R2: A role includes both responsibilities (the service interface) when an agent is taken as a server and rights (the request interface) when the agent is taken as a client.
- R3: Roles can be interface-roles.
- R4: Roles can be process-roles. In such roles, what to do, how to do, and what to access are all rigidly specified.
- R5: Roles are taken as media for interactions.
- R6: Playing a role means that the agent is attached to a role. A role can be played by one or more agents at the same time.
- R7: An agent may play one or more roles but can play only one role at a time.
- R8: Roles can be used to support indirect and direct interactions.
- R9: Roles can have hierarchy relationships. Higher level roles can be taken as goals for agents playing roles at lower levels.


Group principles

- G1: A group is a fundamental structure in a collaborative system.
- G2: A group can be created, changed and deleted.
- G3: Before specifying a group, we must specify all the roles in it.
- G4: To form a group is letting agents join the group and play roles. They are named as the members of this group.
- G5: A group can be embedded, i.e., one group may be an object in another (Tanenbaum and van Steen, 2002).
- G6: A group can be overlapped with other groups, i.e., the members may belong to two or more groups (Tanenbaum and van Steen, 2002).
- G7: A group can be public or private (WebBoardTM, 2006).
- G8: A group can be open or closed (Coulouris et al., 2005).

How Do We Support RBC?
Concepts in role-based collaboration

- An **object** is used to express everything in a collaborative system
- A **human user** is a person who is participating in collaboration
- An **agent** is a special object that represents a human user in collaboration
- A **message** is a method to invoke activities of objects
- An **interface** is a list of messages sent to objects in the system or to the system itself
- A **role** is a special object that symbolizes a logged human user in the system, and a role must have an interface.
- A **class** is a template of objects
- A **group** is a set of agents and objects

---

The architecture of a role-based collaborative system
Messages

- Messages are defined by message identification, a receiver and arguments;
  - The receivers can be categorized as objects, classes or groups;
  - The messages can be categorized as all, any and some messages.
- $M_i = <n, v, l, P>$

A role can be defined a set of messages

- $R_i = <n, M_i, M_o>$ where,
  - $n$ is the identification or the name of the role; and
  - $M_i$ and $M_o$ denote sets of message patterns, wherein, $M_i$ expresses the incoming message patterns to the relevant agent or the human user; $M_o$ express different sets of outgoing message patterns to the objects.
A system is a group of sets

\[ \Sigma ::= <C, O, A, M, R, E, G, s_0, H> \]

where

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

The E-CARGO Model

- **Class**  \( C ::= <n, D, F, X> \)
- **Object**  \( O ::= <n, v, \hat{v}> \)
- **Agent**  \( A ::= <n, e, \hat{e}> \)
- **Messages**  \( M ::= <n, v, l, P> \)
- **Role**  \( R ::= <n, M, M> \)
- **Environment**  \( E ::= <n, B> \)
- **Group**  \( G ::= <n, e, \{h\}, \{a\}> \)
RBC: Collaboration with E-CARGO

- All the world’s a stage (e, c, o),
- And all the men and women merely players (a);
- They all have their exits and entrances (g);
- And one man in his time plays many parts (r).

The Realm of RBC
RBC and related areas

Special RBC

- Special RBC means role-based CSCW research.
- In this kind of system, it is mainly concerned with how to support people to cooperate with computers.
- Tasks:
  - To apply the role theory of Social Psychology to CSCW systems.
- Aims:
  - To create concrete artifacts relevant to the role theory of social psychology in CSCW systems.
  - To bridge the gap between developers of CSCW systems and the sociologists.
General RBC

- General RBC is to extend special RBC to the areas such as Human Computer Interaction (HCI), Artificial Intelligence, Software Engineering (Computer Security), Social Psychology (Organizational and Management Theory).
- General RBC considers supporting not only cooperation among people with computers but also the cooperation among the components of a system, among people, and among people and machines.
- Task:
  - To model systems with roles and relevant concepts
- Aims:
  - Improve the efficiency of system development
  - Improve the performance of systems

Role-Based Multi-Agent Systems
Revised agent principles for multi-agent systems

- A1: Agents are special objects that simulate the behavior of people.
- A2: Agents can be created, modified, and deleted.
- A3: Agents are autonomous. They should be able to reply incoming messages and send outgoing messages based on their situations (Etzioni and Weld, 1995; Jennings et al., 1998; Russell and Norvig, 2003).
- A4: Agents are adaptive. They should be able to understand their environment and take actions to change the environment and make it better for them to live (Etzioni and Weld, 1995; Russell and Norvig, 2003).
- A5: Agents are social. They should be able to interact with other agents (Jennings et al., 1998; Russell and Norvig, 2003).
- A6: Agents are collaborative. They may join a group to work for a common goal or quit a group if they do not want to cooperate more.
- A7: Agents are flexible. Not all actions of agents are predicted. They can dynamically choose which actions to invoke, and in what sequence, in response to the state of its environment (Jennings et al., 1998; Russell and Norvig, 2003).
- A8: Agents are mobile. They are able to transport them from one site to another in a system (Etzioni and Weld, 1995).

Revised E-CARGO model

Definition 3: 
\[ a := \langle a, i, \xi, \rho, N_Y, \eta, e_1, w, u \rangle \], where
- \( a \) and \( i \) have the same meanings as those in Definition 2;
- \( \xi \) is a special class that describes the common properties of agents;
- \( \eta \) means a role that the agent is currently playing. If it is empty, then this agent is free;
- \( \rho \) means a set of roles that the agent is potential to play (\( \rho \subseteq \Xi \));
- \( N_Y \) means a set of identifications of groups that the agent belongs to;
- \( \langle \eta, e_1 \rangle \) expresses the processing capacity for an agent, where \( \eta \) expresses how many units of free time it has and \( e_1 \) expresses how much memory space it has. \( \langle e_2 \rangle \) can be reset based on the performance of an agent’s services. Even though the time in one day is 24 hours, a person may have different \( e_1 \) and \( e_2 \), based on their processing capacities including the working efficiency, attitudes, and goals;
- \( w \) expresses the past performance or credits of serving others; and
- \( u \) to expresses the workload of the agent.
Revised E-CARGO model (CNT’D)

Definition 5: role. \( r := \langle i, I, N, N, e, o, R, w \rangle \) where,

- \( i \) is the identification of the role;
- \( I := \langle M_o, M_c \rangle \) denotes a set of messages, wherein, \( M_o \) expresses the incoming messages to the relevant agents, and \( M_c \) expresses a set of outgoing messages or message templates to roles, i.e., \( M_o, M_c \subseteq M \);
- \( N \) is a set of identifications of agents that are playing this role;
- \( N \) is a set of identifications of objects including classes, environments, roles, and groups that can be accessed by the agents playing this role;
- \( e_o \) and \( e_i \) are used to express the processing capacity requirement, where \( e_o \) expresses how many units of free time it requires and \( e_i \) expresses how many units of space it requires. \( <e_o, e_i> \) expresses that an agent must possess at least \( <e_o, e_i> \) to play this role;
- \( R \), the super roles; and
- \( R \), the subordinate roles.

Role-based MAS

- With roles, to form a group of agents to collaborate to complete a definite task, we have the following steps:
  - Define roles required by the task (r);
  - Connect the roles with structures (e, c, o);
  - Design agents based on roles (a); and
  - Release agents to play roles in the group (g).
- To accomplish the above tasks, we need to provide an engine possessing the functions as follows:
  - Role specification;
  - Role registration;
  - Role assignment;
  - Role transfer; and
  - Role dynamics.
Architecture design

- The basic procedure is as follows:
  - Identify roles. Analysts extract roles from the problem descriptions.
  - Specify roles. Designers describe the incoming and outgoing messages for the roles.
  - Specify the role relationships. Designers describe the relationships among the roles, such as classification, promotion, request/service, and conflict.

Agent implementation

- An agent might be a machine, a computer, a robot, a hardware component such as a sensor, or software component such as a process.
- To have agents work: the designers specify enough requests by using the provided rights of the roles.
- The designers should concentrate on specifying roles and the relevant requests and services.
Agent integration

- By People
- By system: autonomic computing or role engine.

Keys:
- Role: : Requests: : Services: : Agent: : Current Role:
- : Active Role: : Connection of Requests and Services.

Role-based MAS: summary

- To develop a role-based agent system, we have the following steps: architecture design, agent implementation, and system integration.

- With Role-Based MAS, a new exciting topic occurs: agent dynamics.
  - Roles are major source of forces to attract agents to enter, work and contribute in an agent world.
Software nature

- The nature of software is its “softness”.
- This “softness” leads to four difficult aspects for software development:
  - absence of a fundamental theory,
  - ease of change,
  - rapid evolution of technologies and
  - very low manufacturing costs.
- It is difficult to understand, measure, transfer and reuse.
- Software developers encounter more problems in software engineering than their colleagues do in other engineering disciplines such as chemical, mechanical, industrial, electrical and electronic engineering.
- It is much harder to build reusable software components than to build reusable hardware ones.
Software is hard

- Software is very simple because a teenager could make software.
- Software is very complex because a team of hundreds of experienced engineers and scientists cannot guarantee their software is bug-free.
- We have a paradox “software is hard” (Knuth) because it is soft.
- To consider both software product (individual intelligence-related) and software development procedures (collective intelligence-related) and give a consistent methodology for the complete lifecycle of a software project.

Why do we need separation of concerns?

- Software is a set of items or objects that form a “configuration” that includes programs, documents and data.
- Specialization is a key factor in the high production rate of the industrial age.
- The current state Software Engineering (SE) is not satisfactory.
- Separation of concerns is one such innovative ideas.
- Some further readings include publications relevant to Agent-Oriented Software Engineering (AOSE).
Why do we separate designs from implementations?

- “Divide and conquer”.
- Communication, planning, modeling, construction, and deployment should be separated for different groups of people.
- No clear boundary between design and implementation, or between modeling and constructions.
  - Conventional SE uses stepwise refinement of procedure structures.
  - OOSE uses stepwise class specialization.
- Ambiguous boundary between design and implementation
- Help designers care little about details.
- Help programmers concentrate on their implementation tasks.
- Help people specialize their capabilities on special skills.

How can design be separated from implementation with roles?

- Role-Based Software Development (RBSD by E-CARGO)
  - Team Management (For people: Role Definition, Role Specification and Role Distributions)
  - Architecture design (For system components: Role Definition, Role Specification and Role Distributions)
  - Object implementation (OOP with C++, Java, C#, ...)
  - System integration (Role-Object Matching and Playing)
Brooks’ Law: adding personnel to a late software project makes it even later because more communications are added.

Team management (cnt’d)

Programming Secretary  Chief Programmer  Backup Programmer

Programmer  Programmer  Programmer
Team management (cnt’d)

Breaking up the Brooks’ Law by team management. Adding personnel does not add more communications.

Architecture design

Keys:
- : Role;
- : Requests;
- : Services;
- : Combination of Requests and Services.
Object implementation

- Breaking up the Brooks’ Law by clarifying implementation jobs. Adding personnel does not add more communications.
- Suppose there are M objects, N programmers making objects and each programmer completes one object at T time units. The time to complete M objects is \( MT/N \). Therefore, adding more (\( \Delta \)) programmers will certainly shorten the time to complete the objects \( MT/(M+\Delta) \).

System integration

Keys:
- Role: ● Requests: ◯ Services: ○ Object: ▶ ◀ Current Role:
- ▶ ◀ Active Role: ○ Connection of Requests and Services:
RBSD: summary

- Software nature requires us to find new ways of software development.
- Clear separation between design and implementation is beneficial to software development.
- Roles and the E-CARGO Model can help implement this separation.
- The future work will be on a role-based software development tool based on the Eclipse platform. This should be a good way to exhibit the usefulness and malleability of the role mechanisms having been developed.

A Formal RBC system: E-CARGO revisited

- The relations among roles
- The relations between roles and agents
- The relations among agents
- Properties of a Role-Based Collaboration (RBC) system
**Definition 1:** role. A role is defined as \( r ::= <n, I, A_r, A_p, A_o, R_o, N_o> \) where,

- \( n \) is the identification of the role;
- \( I ::= <M_in, M_out> \) denotes a set of messages, where \( M_in \) expresses the incoming messages to the relevant agents, and \( M_out \) expresses a set of outgoing messages or message templates to roles, i.e., \( M_in, M_out \subseteq M \);
- \( A_r \) is a set of agents who are currently playing this role;
- \( A_p \) is a set of agents who are potential to play this role;
- \( A_o \) is a set of agents who used to play this role;
- \( R_o \) is a set of roles interrelated with it (see **Definition 16**); and
- \( N_o \) is a set of objects that can be accessed by the agents playing this role.

**Definition 2:** agent. An agent is defined as \( a ::= <n, c_o, s, d_r, R_p, R_o, N_o> \), where

- \( n \) is the identification of the agent;
- \( c_o \) is a special class that describes the common properties of users;
- \( s \) is the qualifications of the agent;
- \( r \) means a role that the agent is currently playing. If it is empty, then this agent is free;
- \( R_p \) means a set of roles that the agent is potentially to play \( (r \not\in a.R_p) \), and
- \( R_o \) means a set of roles that the agent played before; and
- \( N_o \) means a set of groups that the agent belongs to.
The Relations among Roles

Classes and instances

- **Definition 3**: role class. Role classes are instances of \( r \).
  - Note that, for a role class \( r \), \( r.N_o \) is a set of classes.

- **Definition 4**: role instance. A role instance is an instance of a role class.
  - Note that, for a role instance \( r \), \( r.N_o \) is a set of objects, or instances of classes.
  - Roles instances are attached to the agent playing its role class.
Inheritance

**Definition 5:** inheritance relation, super roles and sub roles. Role \( r_i \) is a super role of role \( r_j \) if \( r_i \) possesses all the properties of \( r_j \). Vise versa, \( r_i \) is a sub role of \( r_j \). An inheritance relation denoted as \( \Omega \) is a set of tuples of roles \( \langle r_i, r_j \rangle \), where, \( r_i \) is a super role of \( r_i \) and \( r_j \) is a sub role of \( r_i \). Here, “inheritance” is similar to the inheritance concept of object-orientation.

**Definition 6:** root role and leaf role. Role \( r_i \) is called a root role if it has no super roles. Role \( r_i \) is called a leaf role if it has no sub roles.

---

**Super and sub roles**

![Diagram of roles hierarchy](image)

**Figure 1.** Super Roles and Sub Roles

Note: CS means Computer Science
Related definitions for inheritance

**Definition 7:** message types. In RBC, a message can be three types: any, all, and some. Suppose \( r \) is a non-leaf role, \( r_0, r_1, \ldots, r_n \) are leaf roles whose super role is \( r \). *Any-messages* to \( r \) should be sent to an agent that plays \( r_0, r_1, \ldots, r_n \). *All-messages* should be sent to all the agents that play \( r_0, r_1, \ldots, r_n \). *Some-messages* should be sent to some (>1) agents that play \( r_0, r_1, \ldots, r_n \).

**Definition 8:** successful message sending. When a message is sent to a role, it is successful if the role covers the message pattern and it finally finds its receivers, i.e., agents.

Role Promotion

**Definition 9:** role promotion, lower role and upper role. Role \( r_i \) is an upper role of role \( r_j \) if \( r_i \) must be played by agent \( a \) before \( r_j \) is assigned to \( a \). Vise versa, \( r_i \) is called a lower role of \( r_j \). A promotion relation denoted as \( \mathcal{A} \) is a set of tuples of roles \(< r_i, r_j >\), where \( r_i \) is a lower role of \( r_j \) and \( r_j \) an upper role of \( r_i \).
Promotion

![Diagram of Promotion Relations]

**Figure 2. Examples of Promotion Relations**

Report-to

**Definition 10:** *report-to relation, supervisor role and supervisee role.* Role $r_j$ is a *supervisor role* of role $r_i$ if role $r_i$ must respond to the requests from role $r_j$ in a time limit set by $r_j$. Vice versa, $r_i$ is a *supervisee role* of $r_j$. A *report-to relation* denoted as $\Delta$ is a set of tuples of roles $<r_i, r_j>$, where, $r_i$ is a supervisee role of $r_j$ and $r_j$ is a supervisor role of $r_i$. 
Definition 11: request relation, service role and request role. Role \( r_i \) is called a request role of role \( r_j \) if \( r_j \) provides the services requested by \( r_i \), i.e., \( r_i.M_{\text{req}} \subseteq r_j.M_{\text{ser}} \). Vise versa, \( r_j \) is called a service role of \( r_i \). A request relation denoted as \( \Theta \) is a set of tuples \( <r_i, r_j> \), where \( r_i \) is a request role of \( r_j \) and \( r_j \) is a service role of \( r_i \), i.e., \( <r_i, r_j> \in \Theta \) if \( r_i.M_{\text{req}} \subseteq r_j.M_{\text{ser}} \).

---

**Figure 3. A Request Relation**
Competition

**Definition 12:** competition relation and competitor role. Role \( r_i \) is called a **competitor role** of role \( r_j \) if roles \( r_i \) and \( r_j \) have the same request role or the same upper role. A **competition relation** denoted as \( \mathcal{E} \) is a set of tuples of roles \(<r_i, r_j>\), where, \( r_i \) and \( r_j \) are competitor roles of each other. More exactly, \(<r_i, r_j>\in \mathcal{E} \) if \( \exists r_k \) \( \exists \left( <r_i, r_j>\in \Theta \land <r_k, r_j>\in \Theta \lor ( <r_i, r_k>\in \mathcal{A} \land <r_k, r_j>\in \mathcal{A} ) \right) \).

Peer

**Definition 13:** peer relation and peer role. Role \( r_i \) is a **peer role** of role \( r_j \) if \( r_i \) and \( r_j \) have the same supervisor role. A **peer relation** denoted as \( \mathcal{O} \) is a set of tuples of roles \(<r_i, r_j>\), where, \( r_i \) and \( r_j \) are peer roles of each other. More exactly, \(<r_i, r_j>\in \mathcal{O} \) if \( \exists r_k \) \( \exists \left( <r_i, r_k>\in \mathcal{A} \land <r_j, r_k>\in \mathcal{A} \right) \).
Conflict

Definition 14: conflict relation, conflict roles. Roles $r_i$ and $r_j$ are conflict if one agent cannot play them together. $r_i$ is called a conflict role of $r_j$ and vice versa. A conflict relation denoted as $\Sigma$ is a set of tuples $<r_i, r_j>$, where $r_i$, $r_j$ are conflict roles of each other. More exactly, $<r_i, r_j> \in \Sigma \rightarrow \forall a \in A, r_i, r_j \in R, (\neg (r_i \in a.R_a \land r_j \in a.R_b))$.

Summary

As a summary, $T$ is used to express all the relations among roles in an RBC system. Suppose $r_i$, $r_j \in R$ and $r_i \neq r_j$, $T := <\Omega, A, \Theta, \in, \Xi, \Sigma>$, where,

- $\Omega$: an inheritance relation. $<r_i, r_j> \in \Omega$ if $r_i$ inherits from $r_j$.
- $A$: a promotion relation. $<r_i, r_j> \in A$ if $r_i$ is a lower role of $r_j$ and $r_j$ is an upper role of $r_i$.
- $\Theta$: a report-to relations. $<r_i, r_j> \in \Theta$ if $r_j$ is a supervisee role of $r_i$ and $r_i$ is a supervisor role of $r_j$.
- $\Theta$: a request relation. $<r_i, r_j> \in \Theta$ if $r_i, I.M_{aui} \subseteq r_i, I.M_{ai}$.
- $\Xi$: a competition relation. $<r_i, r_j> \in \Xi$ if $\exists r_k \in (r_i, r_j) \wedge (r_i, r_k) \in A \lor (r_j, r_k) \in A$.
- $\Theta$: a peer relation. $<r_i, r_j> \in \Theta$ if $\exists r_k \in (r_i, r_j) \wedge (r_i, r_k) \in A \lor (r_j, r_k) \in A$.
- $\Xi$: a conflict relation. $<r_i, r_j> \in \Xi$ if $r_i$ and $r_j$ are conflict.
Properties of the relations

Property 1: irreflexive. \( \forall \langle r, r \rangle \in R(\langle \text{cnt}, r \rangle \in \Omega) \land (\langle r, r \rangle \notin \Delta) \land (\langle r, r \rangle \notin \Omega) \land (\langle r, r \rangle \notin \Theta) \land (\langle r, r \rangle \notin \emptyset) \land (\langle r, r \rangle \notin \Xi) \).

Property 2: transitive.
- \( \forall r, r_1, r_2 \in R(\langle r, r_1 \rangle \in \Omega \land \langle r_1, r_2 \rangle \in \Omega \rightarrow \langle r, r_2 \rangle \in \Omega) \);
- \( \forall r, r_1, r_2 \in R(\langle r, r_1 \rangle \in \Delta \land \langle r_1, r_2 \rangle \in \Delta \rightarrow \langle r, r_2 \rangle \in \Delta) \);
- \( \forall r, r_1, r_2 \in R(\langle r, r_1 \rangle \in \emptyset \land \langle r_1, r_2 \rangle \in \emptyset \rightarrow \langle r, r_2 \rangle \in \emptyset) \); and
- \( \forall r, r_1, r_2 \in R(\langle r, r_1 \rangle \in \Theta \land \langle r_1, r_2 \rangle \in \Theta \rightarrow \langle r, r_2 \rangle \in \Theta) \).

Properties (CNT’D)

Property 3: symmetrical.
- \( \forall r, r_1 \in R(\langle r, r_1 \rangle \in \emptyset) \rightarrow (\langle r, r_1 \rangle \in \emptyset) \);
- \( \forall r, r_1 \in R(\langle r, r_1 \rangle \in \emptyset) \rightarrow (\langle r, r_1 \rangle \in \emptyset) \); and
- \( \forall r, r_1 \in R(\langle r, r_1 \rangle \in \Xi) \rightarrow (\langle r, r_1 \rangle \in \Xi) \).

Property 4: noncircular. There are not circles in inheritance, request, promotion, and report-to relations, i.e., \( \exists r_0, r_1, \ldots, r_n \in R, \langle r_0, r_1 \rangle (r_1 \neq r_2, \ldots, r_{n-1} \rho (r_n = \Omega, \Lambda) \lor \Theta) (i = 0, 1, \ldots, n-1) \forall (r_n = r_0). \)
Role graph

**Definition 15:** *role graph.* A role net is a directed graph [13] formed by all the relations of $T,$ denoted as $T.G,$ i.e., $T.G := \langle R, T \rangle,$ where, $R$ are the node set and $T$ are the edge set.

**Definition 16:** *interrelated roles.* Roles $r_i$ and $r_j$ are interrelated if $<r_i, r_j>$ belongs to the role graph, i.e. $<r_i, r_j> \in (\Omega \cup A \cup \bar{A} \cup \Theta \cup \kappa \cup \hat{0} \cup \Xi).$
Current/potential/past role/agent

**Definition 17**: *current role/agent*. Role \( \rho \) is the current role of agent \( a \) if \( a \) is currently playing \( \rho \), i.e., \( \rho = a\,\hat{a} \), at the same time, \( a \) is called as a current agent of role \( \rho \), i.e., \( a \in \rho, A_c \).

**Definition 18**: *potential role/agent*. Role \( \rho \) is a potential role of agent \( a \) if \( a \) is qualified to play but not currently playing this role, i.e., \( \rho \in a\,\mathcal{R}_p \), at the same time, \( a \) is called as a potential agent of \( \rho \), i.e., \( a \in \rho, A_p \).

**Definition 19**: *past role/agent*. Role \( \rho \) is a past role of agent \( a \) if \( a \) used to play \( \rho \) but \( \rho \) is neither a current nor a potential role, i.e., \( \rho \in a\,\mathcal{R}_c \), at the same time, \( a \) is called as a past agent of \( \rho \), i.e., \( a \in \rho, A_c \).

---

Apply-for and approve

**Definition 20**: *apply-for (a, \( \rho \))*. The predicate is to have agent \( a \) apply for role \( \rho \).

After the predicate *apply-for (a, \( \rho \))* is received, the system should check if \( a \) is qualified to play \( \rho \) based on the qualifications of it.

**Definition 21**: *approve(a, \( \rho \))*. It is a predicate that approves role \( \rho \) for agent \( a \). This event occurs when *apply-for (a, \( \rho \))* has been issued and the agent is evaluated to be qualified to play role \( \rho \). After this predicate is executed, \( \rho \) is added to the potential role set of agent \( a \), i.e., \( \rho \in a\,\mathcal{R}_p \).
Transfer, dispatch and reply

**Definition 23:** transfer\((a, r_i, r_f)\). It is a predicate that agent \(a\) transfers its current role from \(r_i\) to \(r_f\). After it is executed, \(r_f\) is assigned to the current role of agent \(a\) and \(r_i\) is put back to the potential role set, i.e., \((a.r_i = r_f) \land (a.R_r = (a.R_r \cup \{r_i\}) \setminus \{r_f\})\). It can be executed only when the system is still workable role transfer occurs, i.e., all the roles have enough current agents.

**Definition 24:** dispatch\((r, a, m)\). It is a predicate that role \(r\) dispatches message \(m\) to agent \(a\). After it is executed, \(a\) does what \(m\) asks for.

**Definition 25:** reply\((a, r, p)\). It is a predicate that agent \(a\) replies an object \(p\) to role \(r\). After it is executed, \(r\) completes one service and responds to its request role with object \(p\).

Relations among agents
Collaborators

Definition 26: collaborator. Agents $a$ and $b$ are collaborators if their roles are interrelated, $\exists \tau$, $\tau \in a.\mathcal{R} \cap b.\mathcal{R}$ $\exists <\tau, \eta> \in (\mathcal{Q} \cup \mathcal{I} \cup \mathcal{A} \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{E} \cup \mathcal{O} \cup \mathcal{Z})$.

Definition 27: current collaborator. Agents $a$ and agent $b$ are current collaborators if they are currently playing interrelated roles or $a.\tau$ and $b.\tau$ are interrelated, i.e., $<a.\tau, b.\tau> \in (\mathcal{Q} \cup \mathcal{I} \cup \mathcal{A} \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{E} \cup \mathcal{O} \cup \mathcal{Z})$.

Definition 28: potential collaborator. Agents $a$ and $b$ are potential collaborators if they are not current collaborators but there are two interrelated roles in the intersection of their repository role sets, i.e., $\exists \tau, \eta <\tau, \eta> \in (\mathcal{Q} \cup \mathcal{I} \cup \mathcal{A} \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{E} \cup \mathcal{O} \cup \mathcal{Z}) \land <a.\tau, b.\tau> \in (\mathcal{Q} \cup \mathcal{I} \cup \mathcal{A} \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{E} \cup \mathcal{O} \cup \mathcal{Z})$.

Peers

Definition 29: peer. Agents $a$ and $b$ are peers if they play the same role, i.e., $a.\mathcal{R} \cap b.\mathcal{R} \neq \Phi$.

Definition 30: current peer. Agents $a$ and $b$ are current peers if they have the same current role, i.e., $a.\tau = b.\tau$.

Definition 31: potential peer. Agents $a$ and $b$ are potential peers if $a.\mathcal{R} \cap b.\mathcal{R} \neq \Phi \land a.\tau \neq b.\tau$. 

---

100
Competitors

**Definition 32:** competitors. Agents $a$ and $b$ are competitors if their repository role sets contain competition roles, i.e., $\exists r, \xi \in a.\mathcal{R} \cap b.\mathcal{R} \ni \langle r, \xi \rangle \in \mathcal{C}$.

**Definition 33:** current competitor. Agents $a$ and $b$ are current competitors if their current roles are competitor roles, i.e., $\langle a.r, b.r \rangle \in \mathcal{C}$.

**Definition 34:** potential competitor. Agents $a$ and $b$ are potential competitors if $\exists r, \xi \in a.\mathcal{R} \cap b.\mathcal{R} \ni \langle r, \xi \rangle \in \mathcal{C} \land \langle a.r, b.r \rangle \notin \mathcal{C}$.

Client/server

**Definition 35:** client and server. Agent $a$ is called a client of agent $b$ and $b$ is called a server of $a$ if $\exists r, \xi \in a.\mathcal{R} \cap b.\mathcal{R} \ni \langle r, \xi \rangle \in \emptyset$.

**Definition 36:** current client and server. Agent $a$ is called a current client of agent $b$ and $b$ is called a current server of $a$ if $a$ is currently playing a request role of the role $b$ is currently playing, i.e., $\langle a.r, b.r \rangle \in \emptyset$.

**Definition 37:** potential client and server. Agent $a$ is called a potential client of agent $b$ and $b$ is called a potential server of $a$ if $\exists r, \xi \in a.\mathcal{R} \cap b.\mathcal{R} \ni \langle r, \xi \rangle \in \emptyset \land \langle a.r, b.r \rangle \notin \emptyset$.
Properties of an RBC system

Properties

Figure 4. Collaborators, Competitors, and Peers
Properties (CNT’D)

**Definition 38**: redundancy. A system is redundant if some services are not requested, i.e., $\exists (r_j \in \mathcal{R})$

$$\exists (r_j \in \mathcal{R}) (\bigcup_{i=0}^{n-1} I_{M_i^r} \cap \mathcal{R}_i) \neq \emptyset, \text{ where, } n = |\mathcal{R}|.$$ 

**Definition 39**: insufficiency. A system is insufficient if some requests are not served, i.e., $\exists (r_j \in \mathcal{R})$

$$\exists (r_j \in \mathcal{R}) (\bigcup_{i=0}^{n-1} I_{M_i^r} \cap \mathcal{R}_i) \neq \emptyset, \text{ where, } n = |\mathcal{R}|.$$ 

**Definition 40**: consistency. A system is consistent if all the relations are consistent, i.e., it keeps the Properties 1-4.

The State of Arts and Future of RBC
Current hot applications of role concepts

- Roles in system management
  - Continue the effort or RBAC roles
  - Consider mainly the outgoing messages (rights)
  - To generalize roles into a mechanism in system management

- Roles in agent systems
  - Continue the effort of modeling roles
  - Consider mainly the incoming messages (responsibilities)
  - To generalize roles into a mechanism in agent collaboration and interaction

Roles in General

- Fundamental concepts of roles
- Role specification, expression, presentation and management
- Role relationship
- Role transfer
- Frameworks for role-based systems
- Applications of role-based approaches
Special RBC: CSCW and HCI

- Human-factors related to roles
- Role-based user interface design
- Role-based conflict resolution
- Role-based information sharing
- Role-based view sharing
- Roles in social computing systems
- Evaluation of role-based collaboration

General RBC: Management

- Roles in organizations and management
- Role-based coordination
- Role-based organization
- Role-based process management
- Role-based personal Information management
- Roles in emergency management systems
General RBC: Intelligent Systems

- Roles in multi-agent systems
- Role-based interaction
- Role-based learning and reasoning
- Role-based autonomic computing
- Role-based game design

General RBC: Software Engineering

- Roles as models
- Roles as services
- Roles as programming components
- Role-based software development
- Role-based analysis
- Role-based architecture and design
- Role-based programming
- Role-based testing
- Role-based trusted systems
- Software development as role-based collaboration
- Visualization of role-based collaboration
General RBC: E-Learning

- Role-based e-learning systems
- Role-based adaptive e-learning systems
- Role-based intelligent tutoring systems
- Role analysis in collaborative learning environments
- Student and tutor role playing in e-learning environments
- Role bases in e-learning systems: theory and technology

The Emerged Applications of RBC and E-CARGO

- E-CARGO is an aesthetic acronym. A model name should be simple and beautiful. E-CARGO and RBC are easy to be pronounced.
- Applications
  - Social Computing
  - Human-Computer Interaction
  - Management System
  - Emergency management systems
  - Software Development
  - Computer-Supported Cooperative Work (CSCW)
  - Collaborative Intelligent (Agent) Systems
  - Artificial Intelligence
  - Autonomic Computing
  - Information Personalization
Potential Application Fields of RBC

- Collaboration
- Management
- Virtual organizations
- HCI
- Software engineering
- Autonomic computing
- Information personalization
- Programming
- Operating Systems

The Emerged benefits of RBC (Special)

- Identify the human user “self”;
- Avoid irrelevant interruption;
- Enforce independency by hiding people under roles;
- Encourage people to contribute more;
- Remove ambiguities to overcome expectation conflicts.
- Work with personalized user interfaces;
- Concentrate on a job and decrease possibilities of conflicts for shared resources;
- Transfer roles with the requirement of a group; and
- Provide an advanced cyber-infrastructure for virtual organizations.
Emerged benefits of RBC (General)

- Provide a compromised abstraction level;
- Extract knowledge from raw data
- Simulate the social systems;
- Provide an open architecture;
- Decrease the workload of system administrators;
- Implement separation of concerns;
- Decrease the knowledge space of searching;
- Create dynamics for agents; and
- Regulate the ways of collaboration and interaction among agents.

Potential benefits

- It may change the design of OS
- It may change the design of MIS or OA software such as CA (Computer Association) software
- It may change the way of using computers
- It may change the way of sales of software
- It may change the management of production process
- It may change the industry system architecture
Current challenges

- How to provide a well-defined and well-accepted mechanism to specify role (for General RBC)?
  - Formal and systematic tools
- How to match role players and roles?
  - Formal tools, and role mining technologies
- How to provide an efficient engine to support RBC (for general RBC)?
- How to demonstrate that RBC is better than normal collaboration based computers (for Special RBC)?

Research communities

- IEEE SMC Society Technical Committee of Distributed Intelligent Systems
- Special session or track on IEEE SMC conferences (paper submission deadline: Feb.-April, Conference: mid Oct.)
- Workshop on “Role-Based Collaboration” for CSCW, CTS
- Future: IEEE Int’l Symposium on RBC, IEEE Int’l conference on RBC
Conclusion

- Role-based collaboration is no doubt an interesting topic.
- There are many potential applications of this methodology.
- There are still many challenges open for research.
- They will bring us new achievements in different areas both in academia and industry.

References from the presenter

Additional Readings for Agents