

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

- 孔子曰：“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 properties 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

Roles are a concept that provides high-level abstraction

- ❑ Roles model a perspective on a phenomenon. Roles are a tool for conceptualization. Roles can be acquired and abandoned independently of each other. Roles can be organized in hierarchies, generalized or specialized. Various roles of an object may share common structure and behavior. Roles are bound to an existing object and an object may play several roles at a time. Roles are used to emphasize how entities interact with each other. Roles can be dynamic, involving sequencing, evolution, and role transfer.
- ❑ As a computational thinking concept, roles are a fundamental modeling concept; reflect the evolution of objects; express the interfaces among objects; and facilitate the separation of concerns.

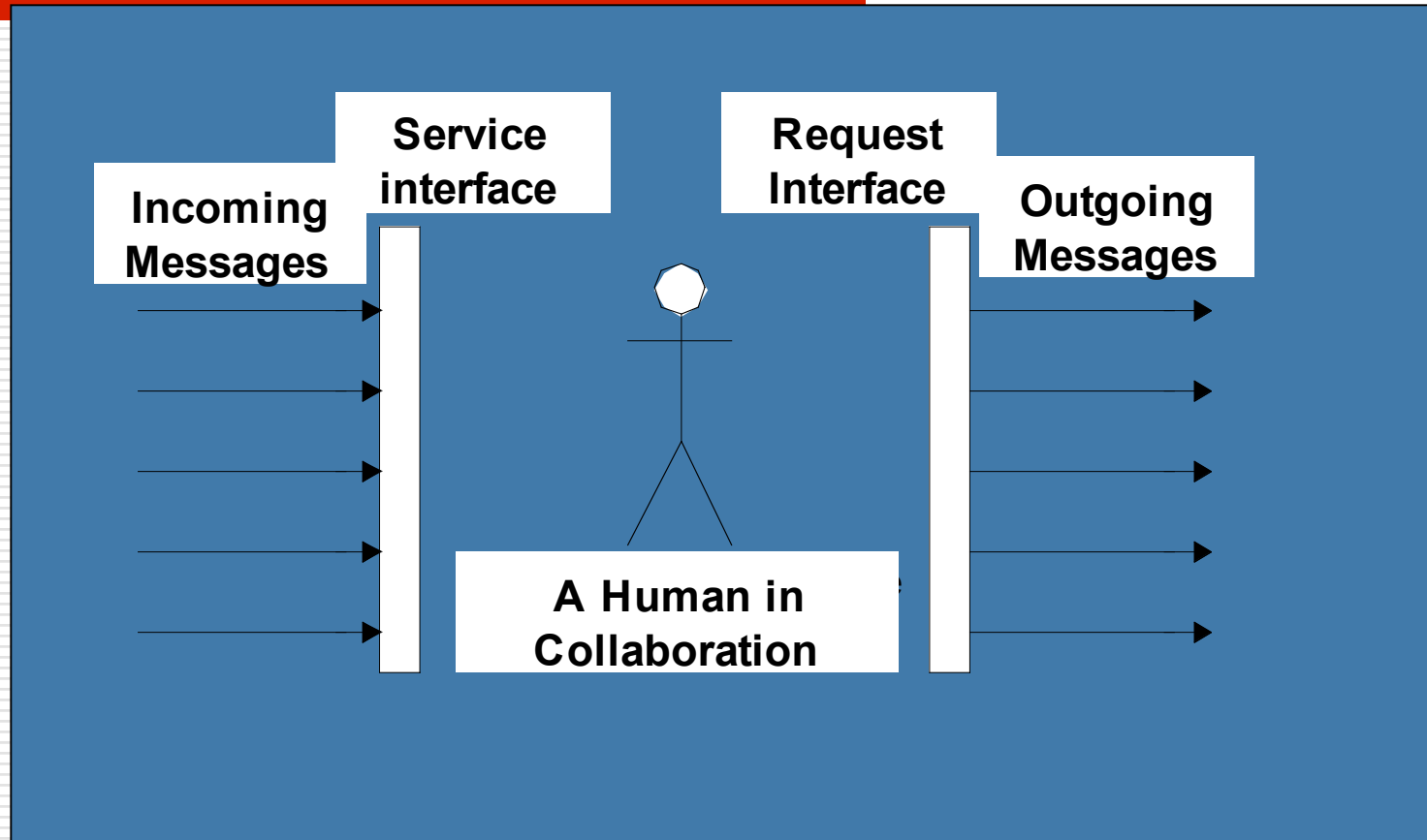
Both rights and responsibilities

- 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

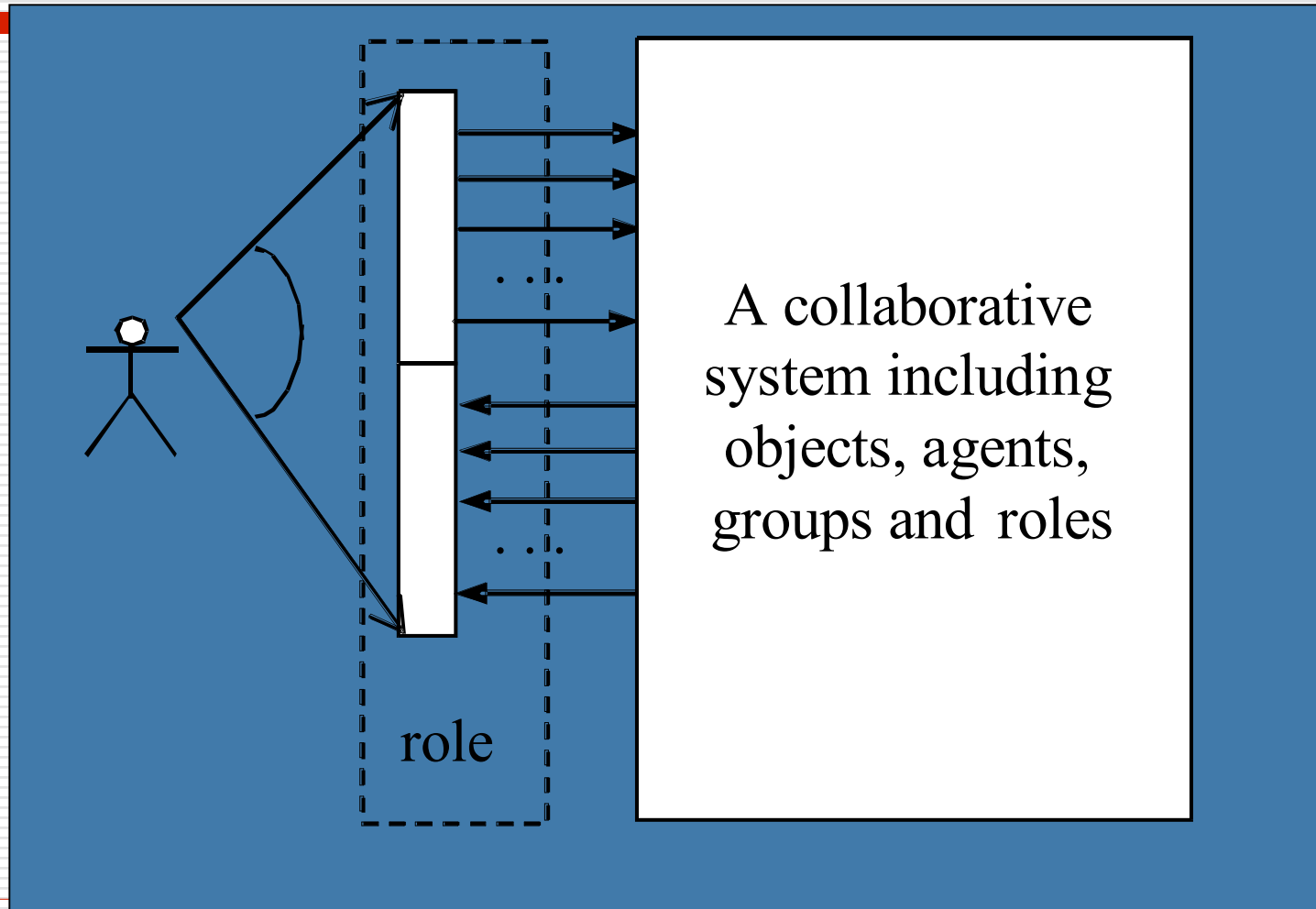
Interfaces or processes

- 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



The occurrence of roles in a system



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 beassigned 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.

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.

What Are The RBC Principles?

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

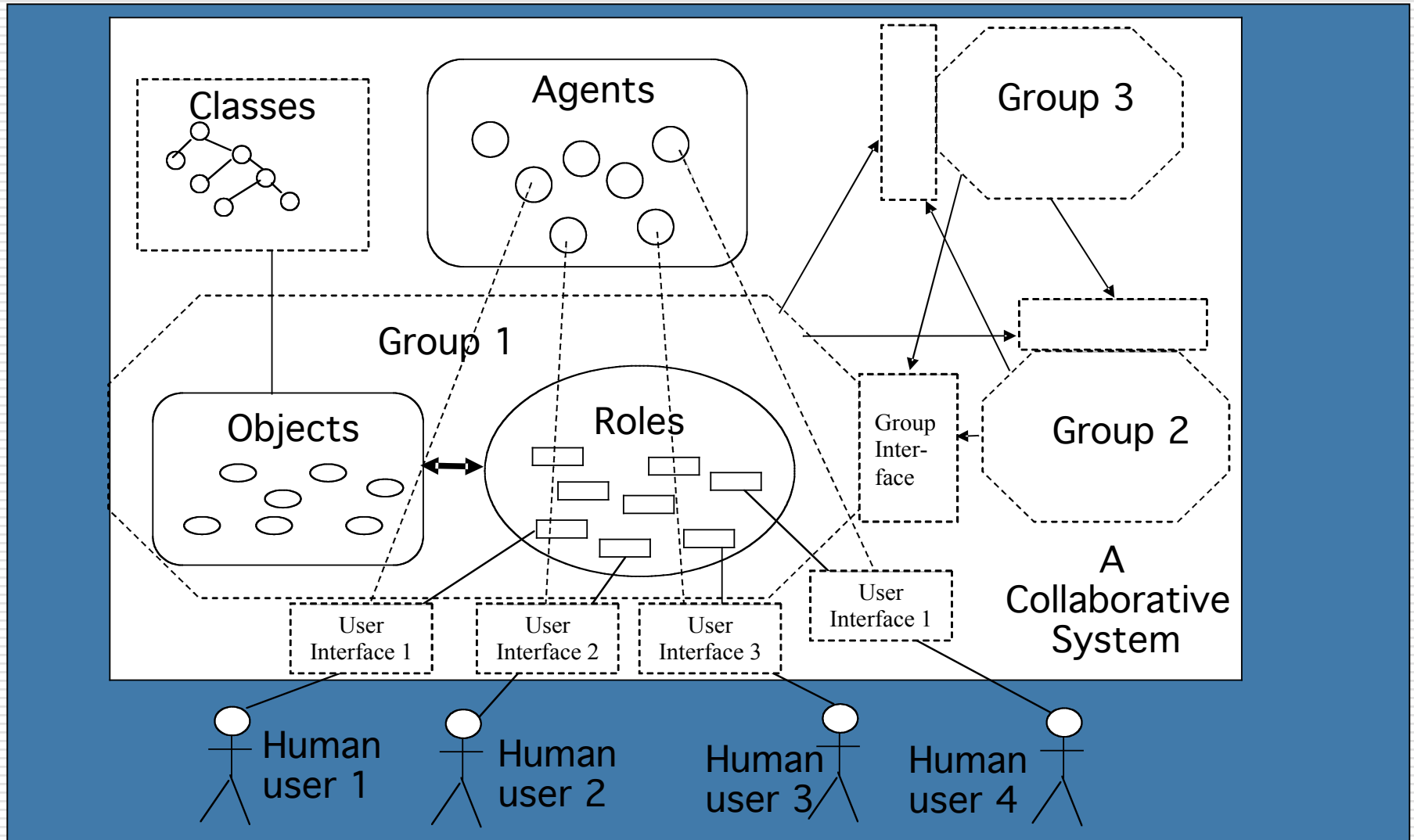
- ❑ G1: A group is a fundamental structure in a collaborative system.
- ❑ G2: A group can be created, changed and deleted.
- ❑ G3: Before specifying a group, we must specify all the roles in it.
- ❑ G4: To form a group is letting agents join the group and play roles. They are named as the members of 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 (WebBoard™, 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.

- $\mathcal{M} ::= \langle n, v, l, \mathcal{P} \rangle$

A role can be defined a set of messages

- $R ::= \langle n, \mathcal{M}_i, \mathcal{M}_o \rangle$ where,
 - n is the identification or the name of the role; and
 - \mathcal{M}_i and \mathcal{M}_o denote sets of message patterns, wherein, \mathcal{M}_i expresses the incoming message patterns to the relevant agent or the human user; \mathcal{M}_o express different sets of outgoing message patterns to the objects.

A system is a group of sets

$\Sigma ::= \langle C, O, \mathcal{A}, \mathcal{M}, \mathcal{R}, \mathcal{E}, \mathcal{G}, s_0, \mathcal{H} \rangle$ where

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

The E-CARGO Model

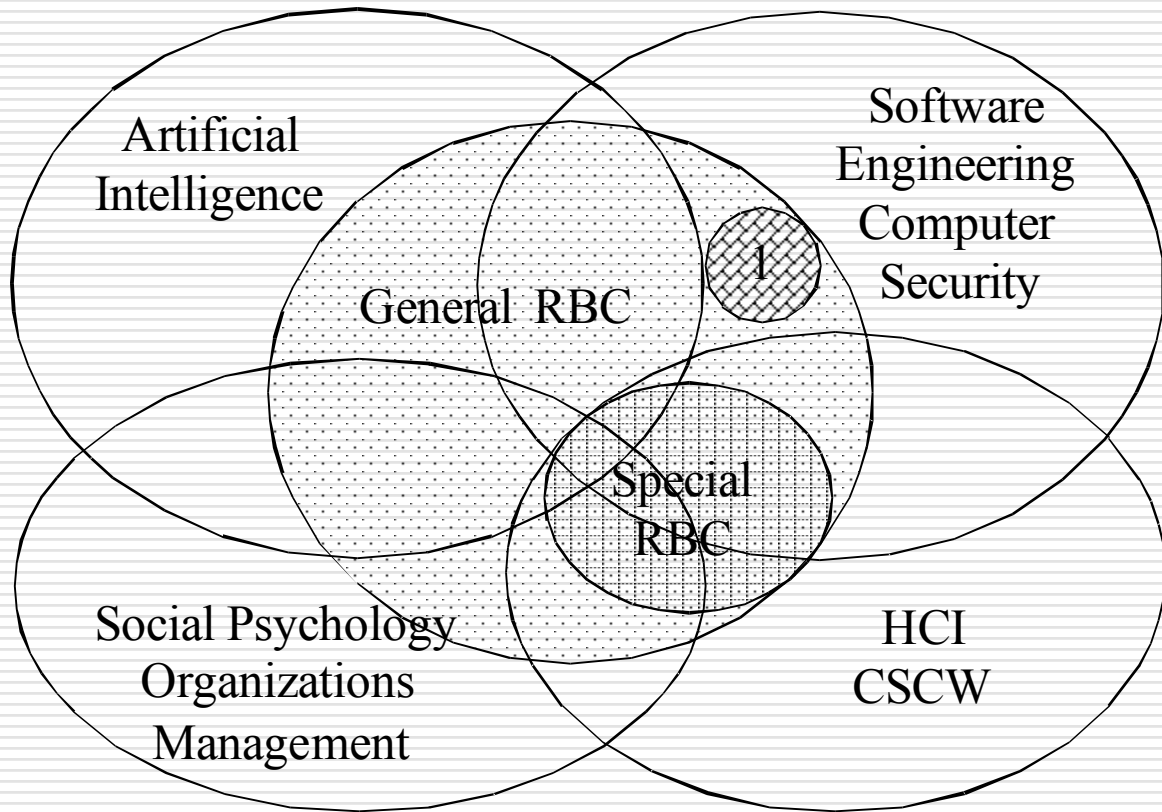
- Class $C ::= \langle n, D, F, N \rangle$
- Object $O ::= \langle n, e, s \rangle$
- Agent $A ::= \langle n, e_a, s \rangle$
- Messages $M ::= \langle n, v, l, P \rangle$
- Role $R ::= \langle n, M_i, M_o \rangle$
- Environment $E ::= \langle n, B \rangle$
- Group $G = \langle n, e, \{h\}, \{a\} \rangle$

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



1: RBAC

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: agent. $a ::= \langle n, c_a, s, r_c, \mathcal{R}_p, \mathcal{N}_g, e_t, e_s, w, u \rangle$, where

- n and s have the same meanings as those in Definition 2;
- c_a is a special class that describes the common properties of agents;
- r_c means a role that the agent is currently playing. If it is empty, then this agent is free;
- \mathcal{R}_p means a set of roles that the agent is potential to play ($r_c \notin a.\mathcal{R}_p$);
- \mathcal{N}_g means a set of identifications of groups that the agent belongs to;
- $\langle e_t, e_s \rangle$ expresses the processing capacity for an agent, where e_t expresses how many units of free time it has and e_s expresses how much memory space it has. $\langle e_t, e_s \rangle$ can be reset based on the performance of an agent's services. Even though the time in one day is 24 hours, a person may have different e_t and e_s 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. $r ::= \langle n, I, \mathcal{N}_a, \mathcal{N}_o, e_t, e_s, \mathcal{R}_m, \mathcal{R}_s, w \rangle$ where,

- n is the identification of the role;
- $I ::= \langle \mathcal{M}_{in}, \mathcal{M}_{out} \rangle$ denotes a set of messages, wherein, \mathcal{M}_{in} expresses the incoming messages to the relevant agents, 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{N}_a is a set of identifications of agents that are playing this role;
- \mathcal{N}_o is a set of identifications of objects including classes, environments, roles, and groups that can be accessed by the agents playing this role;
- e_t and e_s are used to express the processing capacity requirement, where e_t expresses how many units of free time it requires and e_s expresses how many units of space it requires. $\langle e_t, e_s \rangle$ expresses that an agent must possess at least $\langle e_t, e_s \rangle$ to play this role;
- \mathcal{R}_m the super roles; and
- \mathcal{R}_s the subordinate roles.

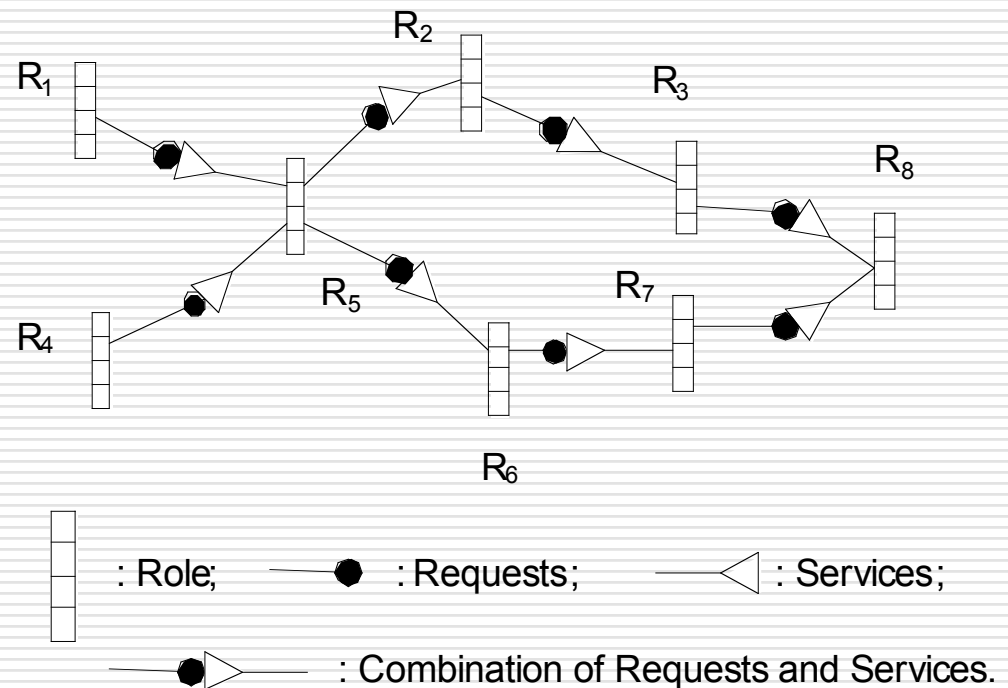
Role-based MAS

- 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

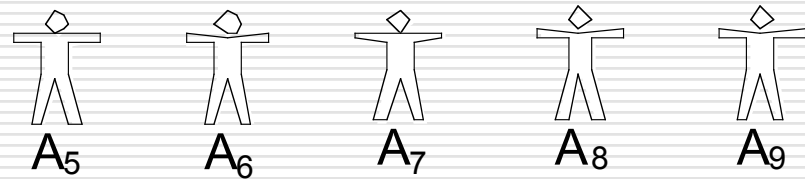
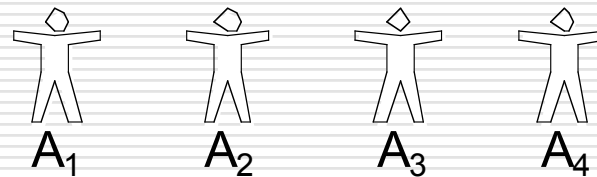
□ The basic procedure is as follows:


- 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

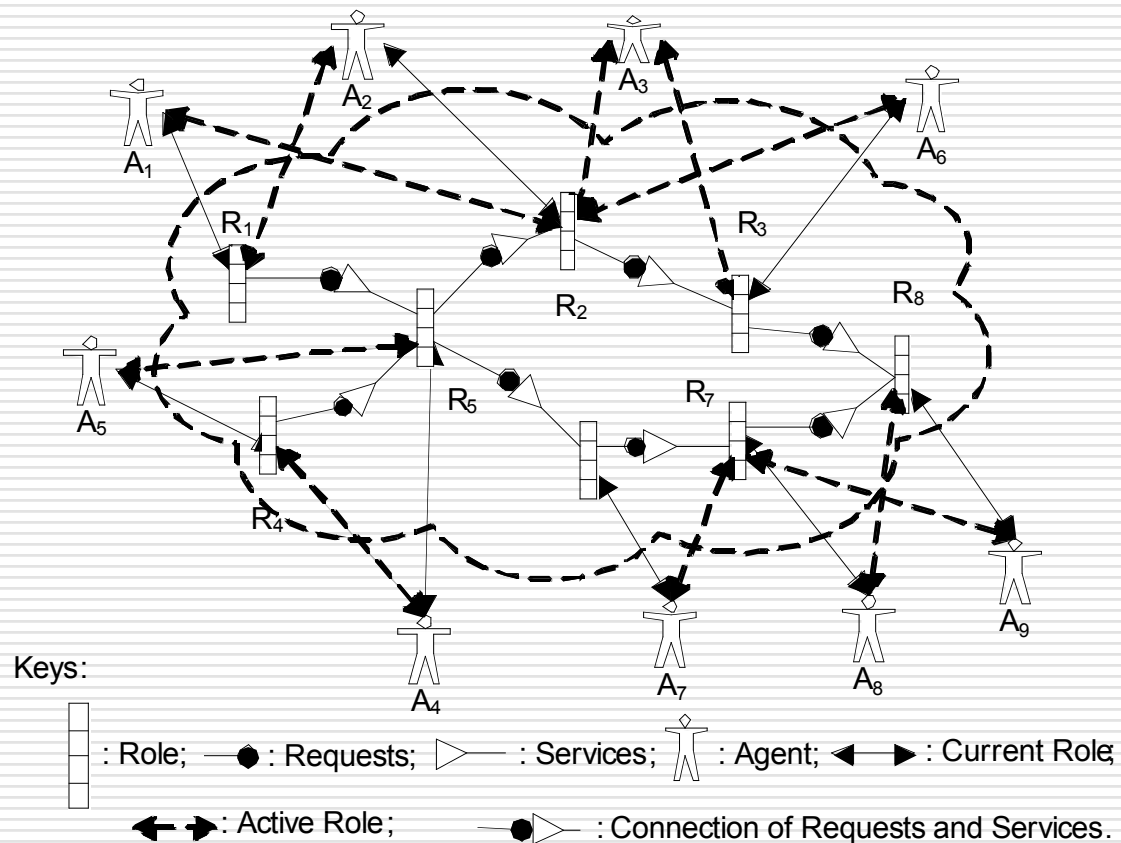
- An agent might be a machine, a computer, a robot, a hardware component such as a sensor, or software component such as a process.
- To have agents work: the designers specify enough requests by using the provided rights of the roles.
- The designers should concentrate on specifying roles and the relevant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



Keys:  : Agent.

Agent integration

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



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.

Role-Based Software Development

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).

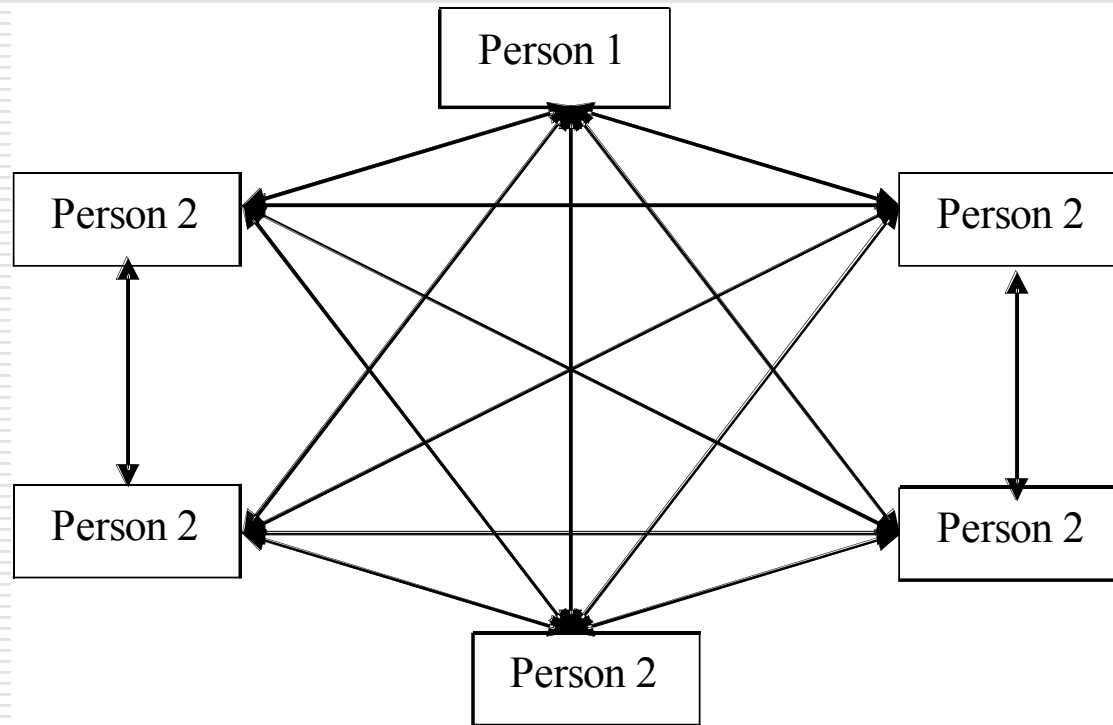
Why do we separate designs from implementations?

- “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?

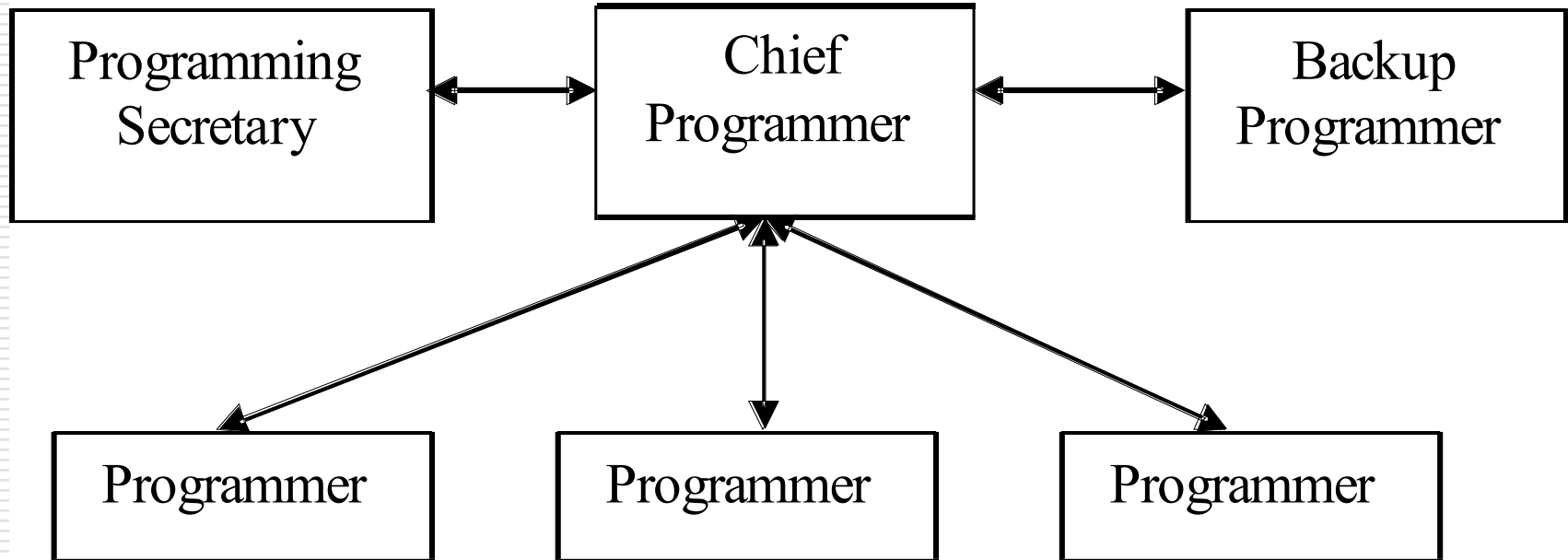
- Role-Based Software Development (RBSD by E-CARGO)
 - Team Management (For people: Role Definition, Role Specification and Role Distributions)
 - Architecture design (For system components: Role Definition, Role Specification and Role Distributions)
 - Object implementation (OOP with C++, Java, C#, ...)
 - System integration (Role-Object Matching and Playing)

Team management

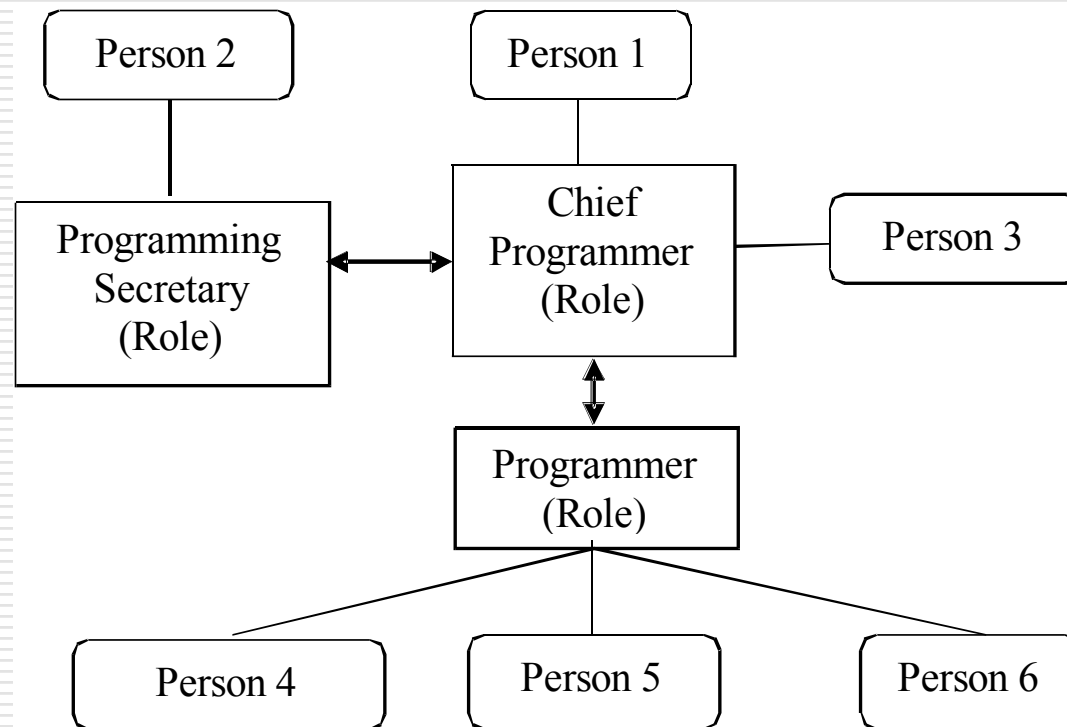


Brooks' Law: adding personnel to a late software project makes it even later because more communications are added.

Team management (cnt'd)

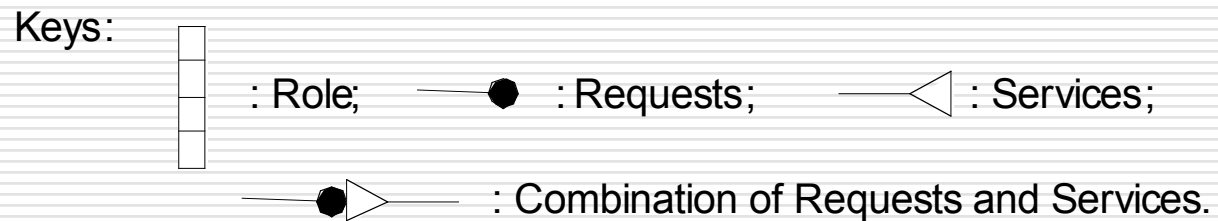
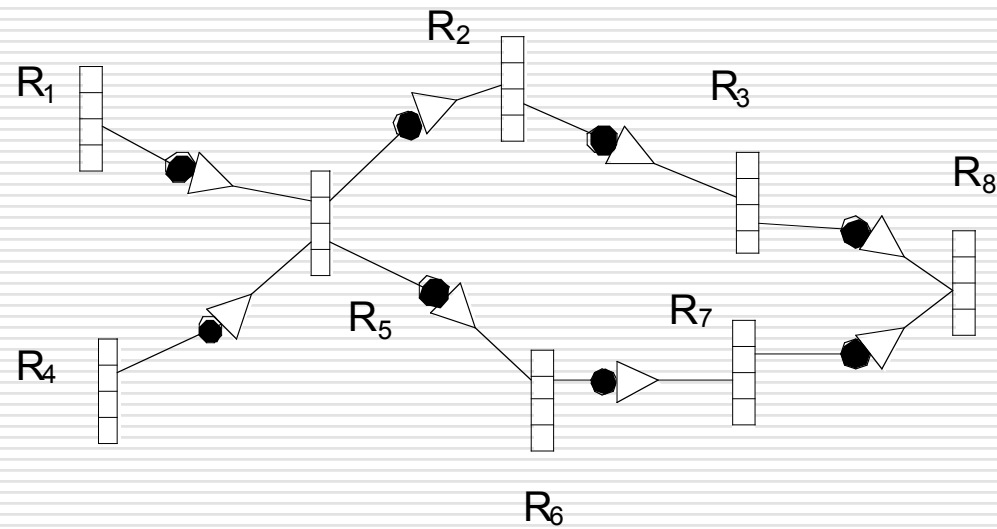
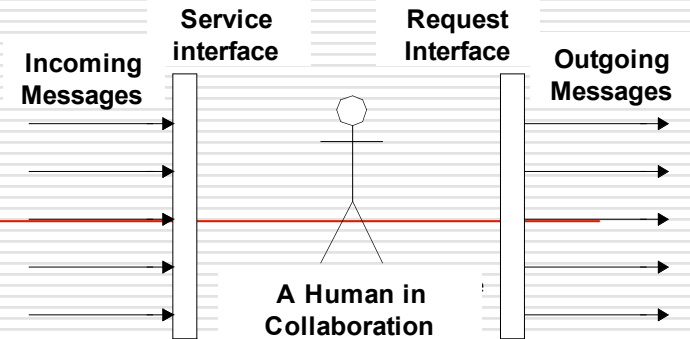


Team management (cnt'd)

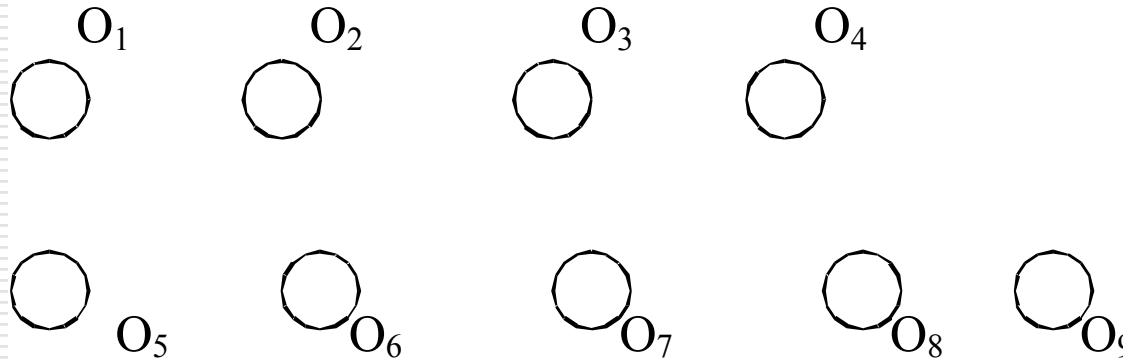


Breaking up the Brooks' Law by team management. Adding personnel does not add more communications.

Architecture design

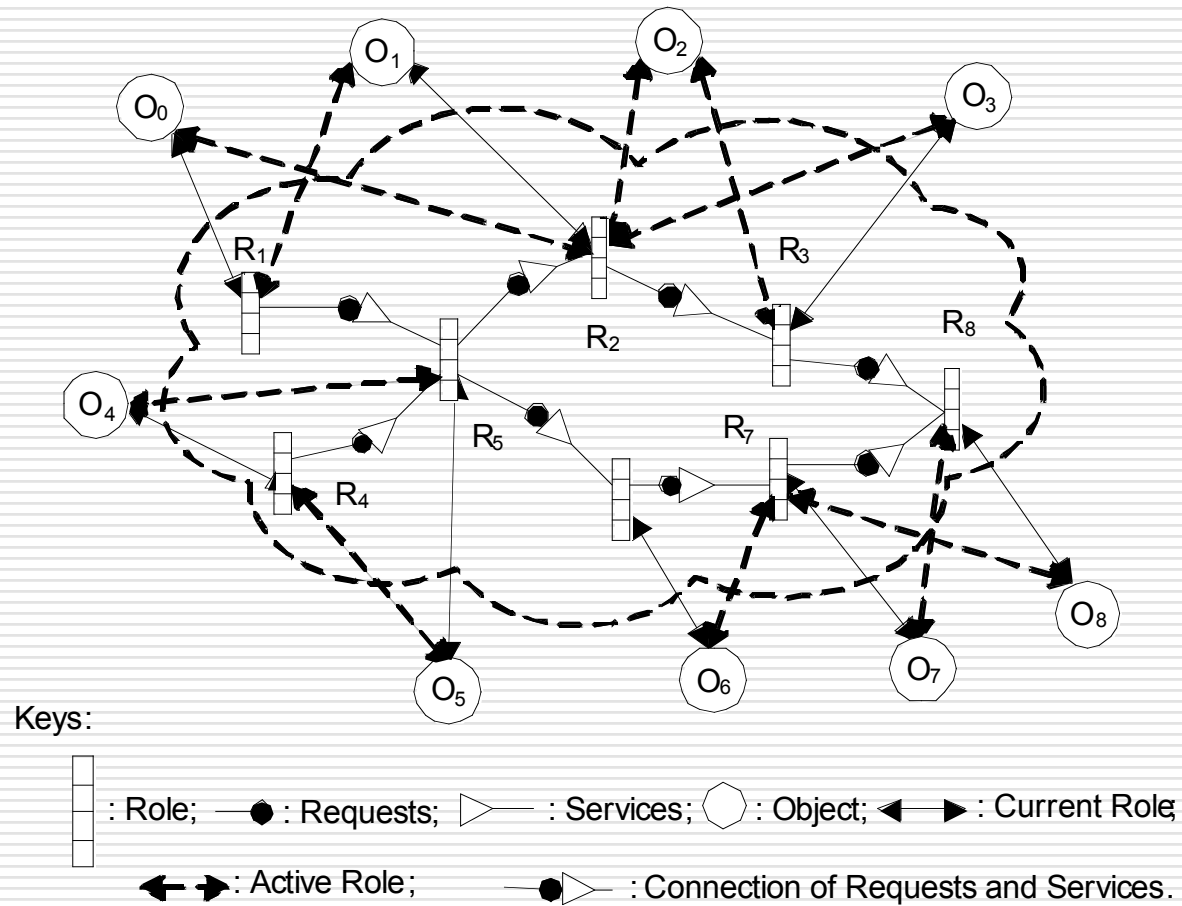


Object implementation



- 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 (Δ) programmers will certainly shorten the time to complete the objects $MT/(M + \Delta)$.

System integration



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

Role

Definition 1: role. A role is defined as $r ::= \langle n, I, \mathcal{A}_c, \mathcal{A}_p, \mathcal{A}_o, \mathcal{R}_x, \mathcal{N}_o \rangle$ where,

- n is the identification of the role;
- $I ::= \langle \mathcal{M}_{in}, \mathcal{M}_{out} \rangle$ denotes a set of messages, 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 are potential to play this role;
- \mathcal{A}_o is a set of agents who used to play this role;
- \mathcal{R}_x is a set of roles interrelated with it (see **Definition 16**); and
- \mathcal{N}_o is a set of objects that can be accessed by the agents playing this role.

Agent

Definition 2: *agent*. An agent is defined as $a ::= \langle n, c_a, s, d, r_c, \mathcal{R}_p, \mathcal{R}_o, \mathcal{N}_g \rangle$, 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;
- r_c means a role that the agent is currently playing. If it is empty, then this agent is free;
- \mathcal{R}_p means a set of roles that the agent is potentially to play ($r_c \notin a. \mathcal{R}_p$); and
- \mathcal{R}_o means a set of roles that the agent played before; and
- \mathcal{N}_g means a set of groups that the agent belongs to.

The Relations among Roles

Classes and instances

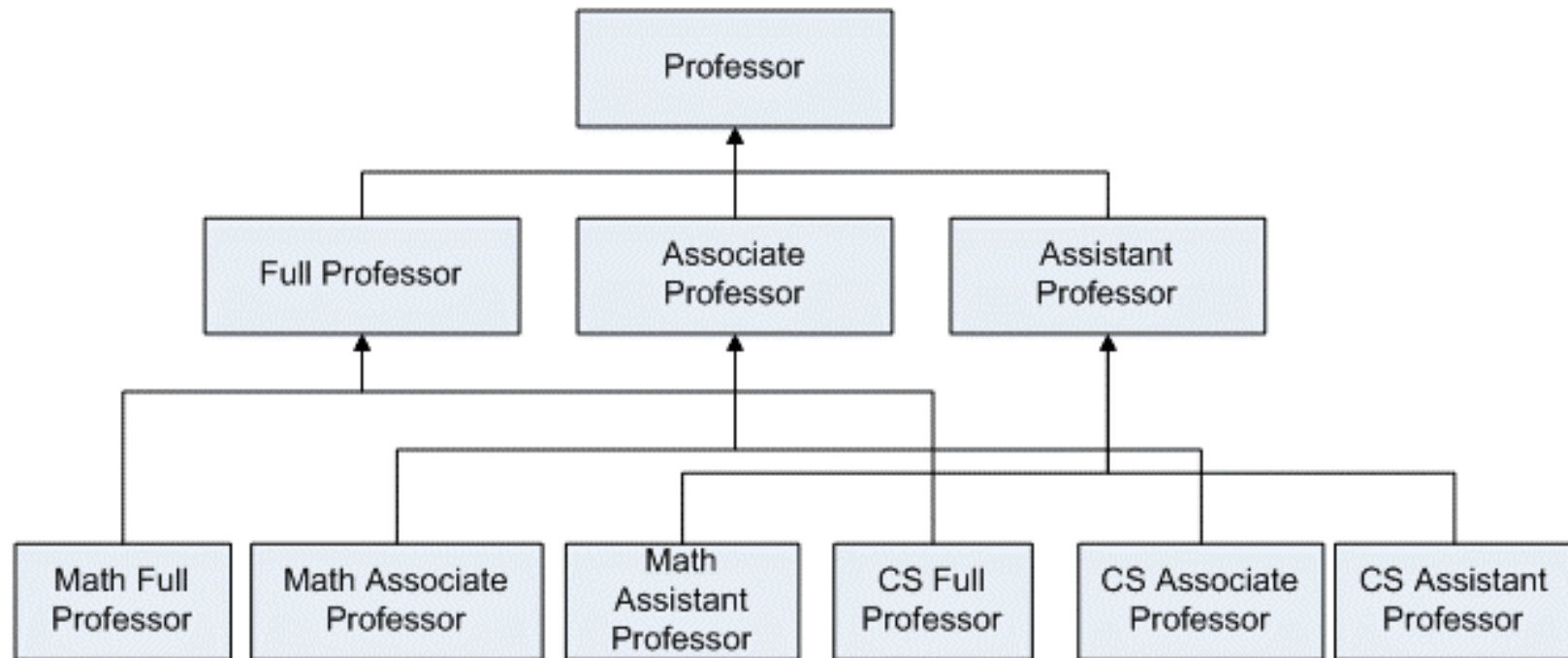
- **Definition 3:** *role class*. Role classes are instances of r .
 - Note that, for a role class r , $r.No$ is a set of classes.
- **Definition 4:** *role instance*. A role instance is an instance of a role class.
 - Note that, for a role instance r , $r.No$ is a set of objects, or instances of classes.
 - 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 . Vice versa, r_j is a *sub role* of r_i . An *inheritance relation* denoted as Ω is a set of tuples of roles $\langle r_i, r_j \rangle$, where, r_i is a super role of r_j 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_j is called a *leaf role* if it has no sub roles.

Super and sub roles



Note: CS means Computer Science

Figure 1. Super Roles and Sub Roles

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, \dots , and r_{n-1} are leaf roles whose super role is r . *Any-messages* to r should be sent to an agent that plays r_0, r_1, \dots , or r_{n-1} . *All-messages* should be sent to all the agents that play r_0, r_1, \dots , and r_{n-1} . *Some-messages* should be sent to some (>1) agents that play r_0, r_1, \dots , or r_{n-1} .

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_j is an *upper role* of role r_i if r_i must be played by agent a before r_j is assigned to a . Vice 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 $\langle r_i, r_j \rangle$, where r_i is a lower role of r_j and r_j an upper role of r_i .

Promotion

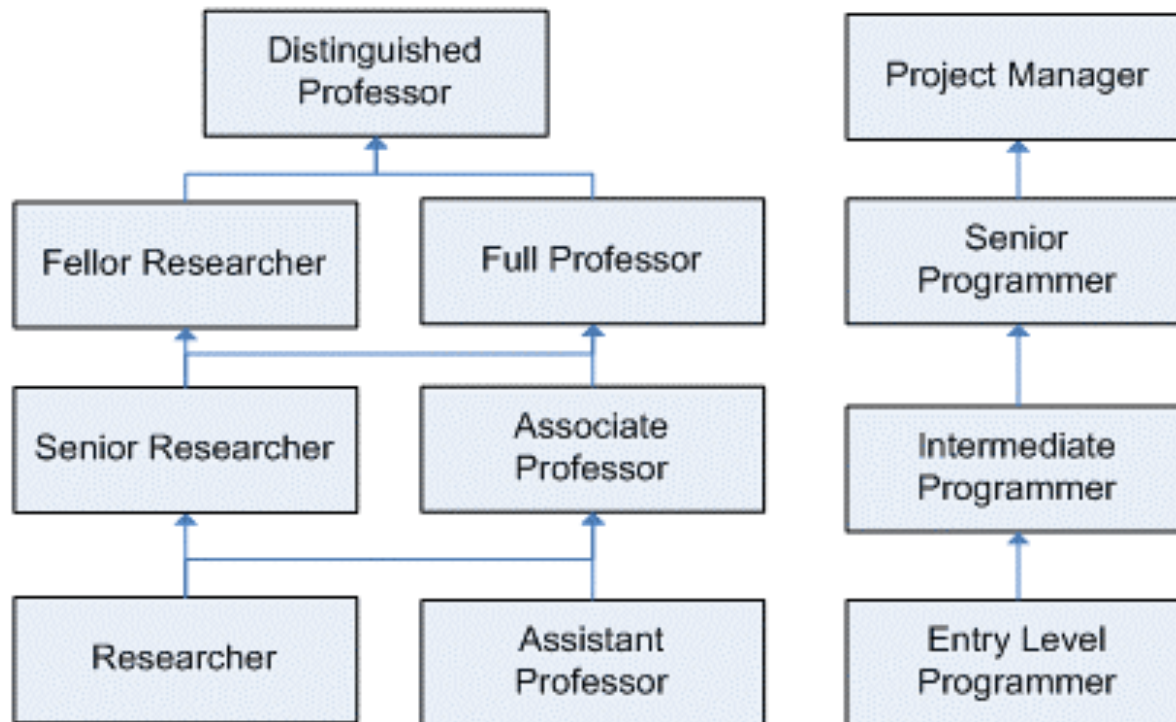


Figure 2. Examples of Promotion Relations

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 \mathcal{A} is a set of tuples of roles $\langle r_i, r_j \rangle$, where, r_i is a supervisee role of r_j and r_j is a supervisor role of r_i .

Request

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., $r_i.I.M_{out} \subseteq r_j.I.M_{in}$. Vice versa, r_j is called a *service role* of r_i . A *request relation* denoted as Θ is a set of tuples $\langle r_i, r_j \rangle$, where r_i is a request role of r_j and r_j is a service role of r_i , i.e., $\langle r_i, r_j \rangle \in \Theta$ if $r_i.I.M_{out} \subseteq r_j.I.M_{in}$.

Request

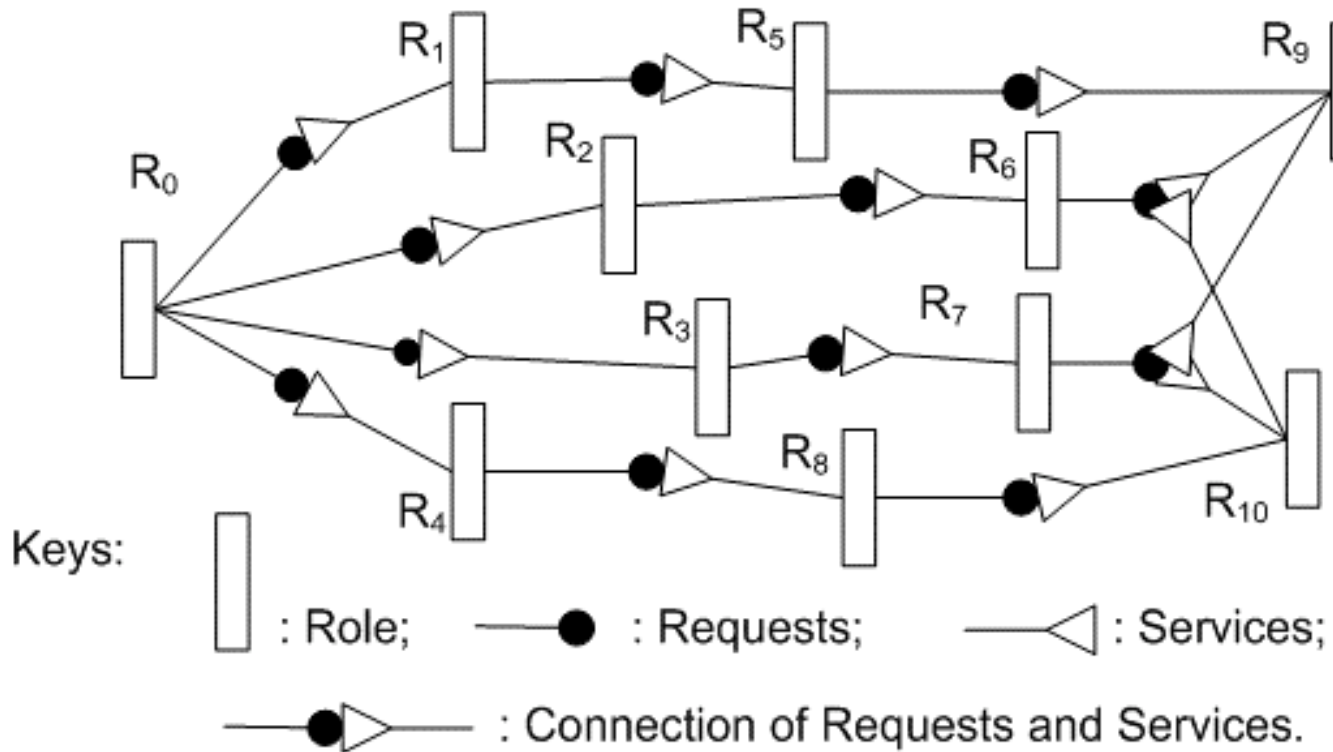


Figure 3. A Request Relation

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 \in is a set of tuples of roles $\langle r_i, r_j \rangle$, where, r_i and r_j are competitor roles of each other. More exactly, $\langle r_i, r_j \rangle \in \in$ if $\exists r_k$
 $\exists (\langle r_k, r_i \rangle \in \Theta \wedge \langle r_k, r_j \rangle \in \Theta) \vee (\langle r_i, r_k \rangle \in \mathcal{A} \wedge \langle r_j, r_k \rangle \in \mathcal{A})$.

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 \diamond is a set of tuples of roles $\langle r_i, r_j \rangle$, where, r_i and r_j are peer roles of each other. More exactly, $\langle r_i, r_j \rangle \in \diamond$ if $\exists r_k \ni (\langle r_i, r_k \rangle \in \mathcal{A} \wedge \langle r_j, r_k \rangle \in \mathcal{A})$.

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 Ξ is a set of tuples $\langle r_i, r_j \rangle$, where r_i, r_j are conflict roles of each other. More exactly, $\langle r_i, r_j \rangle \in \Xi \rightarrow \forall a \in \mathcal{A}, r_i, r_j \in \mathcal{R} (\neg (r_i \in a.R_v \wedge r_j \in a.R_v))$.

Summary

Gamma

Lambda

Omega

Delta

Theta

Euro

Diamond

Xi

As a summary, \mathcal{T} is used to express all the relations among roles in an RBC system. Suppose $r_i, r_j \in \mathcal{R}$ and $r_i \neq r_j$, $\mathcal{T} ::= \langle \Omega, \mathcal{A}, \mathcal{A}, \Theta, \epsilon, \diamond, \Xi \rangle$, where,

- Ω : an inheritance relation. $\langle r_i, r_j \rangle \in \Omega$ if r_i inherits from r_j .
- \mathcal{A} : a promotion relation. $\langle r_i, r_j \rangle \in \mathcal{A}$ if r_i is a lower role of r_j and r_j is a upper role of r_i .
- \mathcal{A} : a report-to relations. $\langle r_i, r_j \rangle \in \mathcal{A}$ if r_i is a supervisee role of r_j and r_j is a supervisor role of r_i .
- Θ : a request relation. $\langle r_i, r_j \rangle \in \Theta$ if $r_i.I.M_{out} \subseteq r_j.I.M_{in}$.
- ϵ : a competition relation. $\langle r_i, r_j \rangle \in \epsilon$ if $\exists r_k \ni (\langle r_i, r_k \rangle \in \Theta \wedge \langle r_j, r_k \rangle \in \Theta) \vee (\langle r_i, r_k \rangle \in \mathcal{A} \wedge \langle r_j, r_k \rangle \in \mathcal{A})$.
- \diamond : a peer relation. $\langle r_i, r_j \rangle \in \diamond$ if $\exists r_k \ni (\langle r_i, r_k \rangle \in \mathcal{A} \wedge \langle r_j, r_k \rangle \in \mathcal{A})$.
- Ξ : a conflict relation. $\langle r_i, r_j \rangle \in \Xi$ if r_i and r_j are conflict.

Properties of the relations

Property 1: irreflexive. $\forall r_i \in \mathcal{R}(\langle r_i, r_i \rangle \notin \Omega) \wedge (\langle r_i, r_i \rangle \notin \Delta)$
 $\wedge (\langle r_i, r_i \rangle \notin \mathcal{A}) \wedge (\langle r_i, r_i \rangle \notin \Theta) \wedge (\langle r_i, r_i \rangle \notin \epsilon) \wedge (\langle r_i, r_i \rangle \notin \diamond)$
 $\wedge (\langle r_i, r_i \rangle \notin \Xi).$

Property 2: transitive.

- $\forall r_i, r_j, r_k \in \mathcal{R}(\langle r_i, r_j \rangle \in \Omega \wedge \langle r_j, r_k \rangle \in \Omega \rightarrow \langle r_i, r_k \rangle \in \Omega);$
- $\forall r_i, r_j, r_k \in \mathcal{R}(\langle r_i, r_j \rangle \in \mathcal{A} \wedge \langle r_j, r_k \rangle \in \mathcal{A} \rightarrow \langle r_i, r_k \rangle \in \mathcal{A});$
- $\forall r_i, r_j, r_k \in \mathcal{R}(\langle r_i, r_j \rangle \in \Delta \wedge \langle r_j, r_k \rangle \in \Delta \rightarrow \langle r_i, r_k \rangle \in \Delta);$
- $\forall r_i, r_j, r_k \in \mathcal{R}(\langle r_i, r_j \rangle \in \epsilon \wedge \langle r_j, r_k \rangle \in \epsilon \rightarrow \langle r_i, r_k \rangle \in \epsilon);$ and
- $\forall r_i, r_j, r_k \in \mathcal{R}(\langle r_i, r_j \rangle \in \diamond \wedge \langle r_j, r_k \rangle \in \diamond \rightarrow \langle r_i, r_k \rangle \in \diamond).$

|

Properties (CNT'D)

Property 3: *symmetrical*.

- $\forall r_i, r_j \in \mathcal{R}(\langle r_i, r_j \rangle \in \mathcal{E}) \rightarrow (\langle r_j, r_i \rangle \in \mathcal{E})$;
- $\forall r_i, r_j \in \mathcal{R}(\langle r_i, r_j \rangle \in \mathcal{D}) \rightarrow (\langle r_j, r_i \rangle \in \mathcal{D})$, and
- $\forall r_i, r_j \in \mathcal{R}(\langle r_i, r_j \rangle \in \mathcal{S}) \rightarrow (\langle r_j, r_i \rangle \in \mathcal{S})$.

Property 4: *noncircular*. There are not circles in inheritance, request, promotion, and report-to relations, i.e., $\neg \exists r_0, r_1, r_2, \dots, r_n \in \mathcal{R}, \langle r_i, r_{i+1} \rangle (r_i \neq r_{i+1}) \in X (X = \Omega, \mathcal{A}, \mathcal{I} \text{ or } \Theta) (i = 0, 1, \dots, n-1) \ni (r_n = r_0)$.

Role graph

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

Definition 16: *interrelated roles.* Roles r_i and r_j are interrelated if $\langle r_i, r_j \rangle$ belongs to the role graph, i.e. $\langle r_i, r_j \rangle \in (\Omega \cup \mathcal{A} \cup \mathcal{B} \cup \Theta \cup \mathcal{E} \cup \mathcal{D} \cup \Sigma)$.

Relations between agents & roles

Current/potential/past role/agent

Definition 17: *current role/agent.* Role r_i is the current role of agent a if a is currently playing r_i , i.e., $r_i = a.r_c$; at the same time, a is called as a current agent of role r_i , i.e., $a \in r_i.A_c$.

Definition 18: *potential role/agent.* Role r_i is a potential role of agent a if a is qualified to play but not currently playing this role, i.e., $r_i \in a.R_p$; at the same time, a is called as a potential agent of r_i , i.e., $a \in r_i.A_p$.

Definition 19: *past role/agent.* Role r_i is a past role of agent a if a used to play r_i but r_i is neither a current nor a potential role, i.e., $r_i \in a.R_o$; at the same time, a is called as a past agent of r_i , i.e., $a \in r_i.A_o$.

Apply-for and approve

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

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

Definition 21: *approve*(a, r_i). It is a predicate that approves role r_i for agent a . This event occurs when *apply-for* (a, r_i) has been issued and the agent is evaluated to be qualified to play role r_i . After this predicate is executed, r_i is added to the potential role set of agent a , i.e., $r_i \in a.R_p$

Transfer, dispatch and reply

Definition 23: $transfer(a, r_i, r_j)$. It is a predicate that agent a transfers its current role from r_i to r_j . After it is executed, r_j is assigned to the current role of agent a and r_i is put back to the potential role set, i.e., $(a.r_c = r_j) \wedge (a.R_p = (a.R_c \cup \{r_i\}) - \{r_j\})$. 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: $dispatch(r_i, a, m)$. It is a predicate that role r_i dispatches message m to agent a . After it is executed, a does what m asks for.

Definition 25: $reply(a, r_i, p)$. It is a predicate that agent a replies an object p to role r_i . After it is executed, r_i 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, $\exists r_i, r_j \in a.R_c \cap b.R_c \ni \langle r_i, r_j \rangle \in (\Omega \cup \mathcal{A} \cup \mathcal{B} \cup \Theta \cup \mathcal{E} \cup \mathcal{F} \cup \mathcal{D} \cup \mathcal{S})$.

Definition 27: current collaborator. Agents a and agent b are current collaborators if they are currently playing interrelated roles or $a.r_c$ and $b.r_c$ are interrelated, i.e., $\langle a.r_c, b.r_c \rangle \in (\Omega \cup \mathcal{A} \cup \mathcal{B} \cup \Theta \cup \mathcal{E} \cup \mathcal{F} \cup \mathcal{D} \cup \mathcal{S})$.

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., $\exists r_i, r_j \in a.R_c \cap b.R_c \ni \langle r_i, r_j \rangle \in (\Omega \cup \mathcal{A} \cup \mathcal{B} \cup \Theta \cup \mathcal{E} \cup \mathcal{F} \cup \mathcal{D} \cup \mathcal{S}) \wedge \langle a.r_c, b.r_c \rangle \notin (\Omega \cup \mathcal{A} \cup \mathcal{B} \cup \Theta \cup \mathcal{E} \cup \mathcal{F} \cup \mathcal{D} \cup \mathcal{S})$.

Peers

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

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

Definition 31: *potential peer*. Agents a and b are potential peers if $a.\mathcal{R}_c \cap b.\mathcal{R}_c \neq \Phi \wedge a.r_c \neq b.r_c$.

Competitors

Definition 32: *competitors.* Agents a and b are competitors if their repository role sets contain competition roles, i.e., $\exists r_i \in a.\mathcal{R}_c \ r_j \in b.\mathcal{R}_c \ \exists \langle r_i, r_j \rangle \in \epsilon$.

Definition 33: *current competitor.* Agents a and b are current competitors if their current roles are competitor roles, i.e., $\langle a.r_c, b.r_c \rangle \in \epsilon$.

Definition 34: *potential competitor.* Agents a and b are potential competitors if $\exists r_i, r_j \in a.\mathcal{R}_c \cap b.\mathcal{R}_c \ \exists \langle r_i, r_j \rangle \in \epsilon \wedge \langle a.r_c, b.r_c \rangle \notin \epsilon$.

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 $\exists r_i, r_j \in a.R_c \cap b.R_c \ni \langle r_i, r_j \rangle \in \Theta$.

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., $\langle a.r_c, b.r_c \rangle \in \Theta$.

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 $\exists r_i, r_j \in a.R_c \cap b.R_c \ni \langle r_i, r_j \rangle \in \Theta \wedge \langle a.r_c, b.r_c \rangle \notin \Theta$.

Properties of an RBC system

Properties

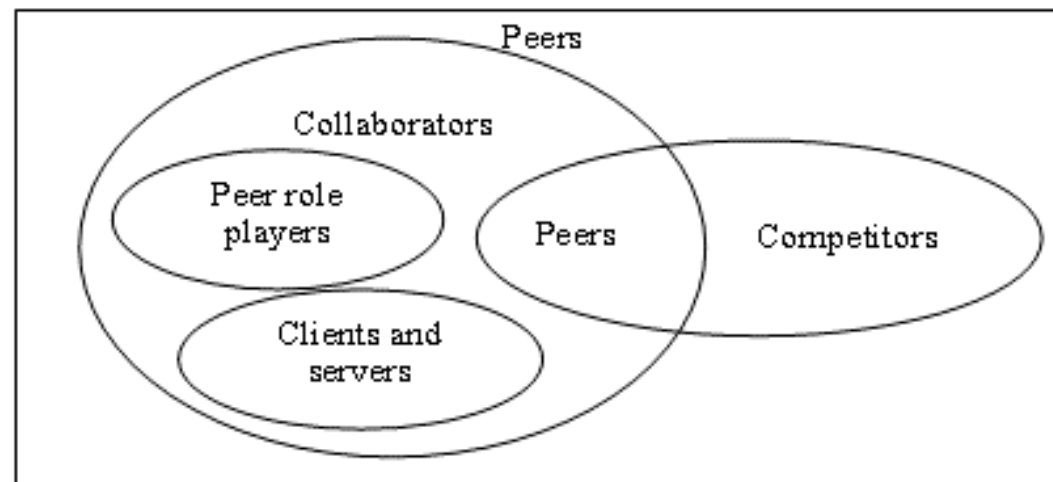


Figure 4. Collaborators, Competitors, and Peers

Properties (CNT'D)

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

$$\exists (r_i.I.\mathcal{M}_{in} \cup_{i=0}^{n-1} r_i.I.\mathcal{M}_{out} (r_i \neq r_j)) \neq \Phi, \text{ where, } n = |\mathcal{R}|.$$

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

$$\exists (r_i.I.\mathcal{M}_{out} \cup_{i=0}^{n-1} r_i.I.\mathcal{M}_{in} (r_i \neq r_j)) \neq \Phi, \text{ where, } n = |\mathcal{R}|.$$

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 on 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.

Emergenced 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

-  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, June 2008, pp.377-398.
-  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.
-  Zhu, H. and Zhou, M.C., "Supporting Software Development with Roles", IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems, vol. 36, no. 6, 2006, pp. 1110-1123.
-  Zhu, H. and Zhou, M.C., "Role-Based Collaboration and its Kernel Mechanisms", IEEE Transactions on Systems, Man and Cybernetics, Part C: Review and Applications, vol. 36, no. 4, July 2006, pp. 578-589.
-  Zhu, H., "Role Mechanisms in Collaborative Systems", International Journal of Production Research, vol. 44, no. 1, Jan. 2006, pp. 181-193.
-  Zhu, H. "Conflict Resolution with Roles in a Collaborative System", International Journal of Intelligent Control and Systems, vol. 10, no. 1, 2005, pp.11-20.
-  Zhu, "Fundamental Issues in the Design of a Role Engine", *Proc. of the 6th Int'l Symposium on Collaborative Technologies and Systems*, Irvine, CA, USA, May 19-23, 2008, pp.
-  Zhu, H., "A Role-Based Approach to Robot Agent Team Design", *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, Oct. 2006, Taipei, China, pp. 4861–4866.
-  Zhu, H. and Zhou, M.C., "The Role Transferability in Emergency Management Systems", the 3rd International Conference on Third International Conference on Information Systems for Crisis Response and Management, Newark, NJ, USA, May 14-17, 2006.
-  Zhu, H., "A Role-Based Architecture for Intelligent Agent Systems", the International Workshop on Distributed Intelligent Systems, Prague, Czech, June 15-17, 2006.
-  Zhu, H. "Encourage Participants' Contributions by Roles", IEEE International Conference on Systems, Man and Cybernetics, Oct. 2005, Hawaii, USA, pp. 1574-1579.

Additional Readings for Agents

-  Jennings, N., and Wooldridge, M. (1996), "Software agents", *IEE Review*, 42(1), 17-20.
-  Jennings, N., Sycara, K, and Wooldridge, M. (1998), "A Roadmap of Agent Research and Development", *Autonomous Agents and Multi-Agent Systems*, 42(1), 7-38.
-  Maes, P. (1994), "Modeling Adaptive Autonomous Agents", *Artificial Life*, 1994, 1(1), 135 - 162.
-  Nwana H.S. (1996), "Software Agents: An overview", *Knowledge Engineering Review*, 11(3), 205-244.
-  Nwana, H.S., Lee, L., and Jennings, N.R. (1996), "Coordination in Software Agent Systems", *BT Technology Journal*, 14(4), 79-89.
-  Odell, J., Nodine, M. and Levy, R. (2005), "A Metamodel for Agents, Roles, and Groups", in Odell, J., Giorgini, P., Müller, J. (Ed.), *Agent-Oriented Software Engineering (AOSE), Lecture Notes on Computer Science*, vol. 3382, Springer, Berlin, 78-92.
-  Odell, J., Van Dyke Parunak, H., and Fleischer, M. (2003), "The Role of Roles in Designing Effective Agent Organizations," in Garcia, A.; Lucena, C.; Zambonelli, F.; Omicini, A.; Castro, J. (Eds.), *Software Engineering for Large-Scale Multi-Agent Systems, Lecture Notes on Computer Science*, vol. 2603, Springer, Berlin, 27-38.
-  Russell, D. and Norvig, P. (2003), *Artificial Intelligence: A Modern Approach (2nd Ed.)*, Pearson Education, Inc., Upper Saddle River, New Jersey.
-  Wooldridge, M. and Jennings, N. (1995), "Intelligent Agents: Theory and Practice", *The Knowledge Engineering Review*, 10(2), 115-152.
-  Wooldridge, M.; Jennings, N. R., and Kinny, D. (2000), "The Gaia Methodology for Agent-Oriented Analysis and Design", *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3), 285-312.
-  Zambonelli, F., Jennings, N. R., and Wooldridge, M. (2003), "Developing Multiagent Systems: The Gaia Methodology", *ACM Trans. on Software Engineering Methodology*, 12(3), 317-370.

Question?