Roles in Information Systems: A Survey

Haibin Zhu, Senior Member, IEEE
Department of Computer Science and Mathematics
Nipissing University
100 College Dr., North Bay, ON P1B 8L7, Canada
email: haibinz@nipissingu.ca

MengChu Zhou, Fellow, IEEE
Electrical and Computer Engineering Department
New Jersey Institute of Technology
University Heights, Newark, NJ 07102, USA
email: zhou@njit.edu

ABSTRACT

Role-based approaches are emerging technologies in information system design and implementation. Roles have been acknowledged and applied in many fields for many years. Considering their increasing importance and applications in the development of various information systems, this paper intends to:

1) Survey the literature relevant to role mechanisms and role-based systems in different fields and point out their motivations and contributions;
2) Classify roles in different contexts as modeling, designing, management, and collaboration;
3) Identify the commonalities and differences among roles in different fields; and
4) Point out the challenges, benefits, and future research areas of role-based systems.

Keywords
Information Systems, Agent Systems, Role, Role Concept, Role Mechanism.

1. INTRODUCTION

From experience, we have learned that many good managerial methodologies are well adopted in developing computer-based systems. Many concepts in people’s lives are successfully introduced into systems. Roles are important in our social lives. They have been applied in China 2000 years ago [97] to help people manage and understand societies.

Role concepts have been applied widely in management, sociology, and psychology for many years [7, 13, 14, 41, 42, 99]. Roles are very useful in modeling the authority, responsibility, functions and interactions associated with manager positions within organizations. Recently, more and more papers that apply roles have been published. Roles are paid more and more attention in different areas relevant to information systems such as modeling [4, 8, 10, 35-39, 47, 49, 50, 56, 67-70, 72-73, 75-77], software design [25, 45, 47, 56, 69, 88, 97], access control [2, 3, 12, 22, 31-33, 46, 65, 76], system administration [54, 61, 75], agent systems [11, 18-20, 28, 30, 59, 60, 66, 84], database systems [21, 62, 71], and Computer-Supported Cooperative Work (CSCW) systems [9, 29, 40, 52, 74, 87, 92, 94, 96].

The first reason why roles are widely applied in information systems is that such systems are tools helping people perform their tasks better and more efficiently. In daily life, working together in harmony requires that everybody fulfill his/her obligations and respect everybody else’s rights. In other words, people are required to respect the social laws of their community [7]. To build a system that can be trusted and applied by people in organizations requires the same rule. Within organizations, individuals fill specific positions (or roles). A position (or role) in this respect represents a specific “seat” that has certain privileges and accompanying responsibilities [13].

When there is no clear role specification, role ambiguity occurs in an organization. It may form conflicting expectations and demands. Role conflicts can occur between an individual’s internal standards and required job (or role) behavior, among time, resources, and capabilities of an individual, and among several roles for the same person, thus leading to incompatible behavior of the person [45]. Role adjustment is required to help people at work cope with ambiguity and conflict in their roles. Based on their experience about a speech-act-based office model, Auramaki et al. state [6]:

“All tasks cannot be predefined and roles evolve. Sometimes the actual role performance depends on a role holder’s skills and capabilities. It is often difficult to assign commitments to the roles without a more detailed knowledge of a role holder’s capacity.”

As a common word, “role” is easily understood and everybody can give examples to describe what a role is. However, as a concept, there is no generally accepted definition [66, 82]. This is also a conclusion we reached after mining the literature on roles.

Although there is a common belief that roles are important concepts, until now no consensus has been reached as to how roles should be represented and integrated into information systems. Roles are granted special meanings in different areas of research, and researchers take their special understanding to support their specific research and development projects. Many different role concepts have been applied in different areas and discussed from different viewpoints. The actual situation of role applications in information systems is definitely in a chaos. There are no clear statements of what roles are in papers too numerous to list in the references. We encounter many difficulties in classifying the relationships of role concepts and mechanisms in different areas of applications. There are not many relations and inheritances among the role mechanisms applied in different studies and applications. Many early information system papers using roles did not refer to the research on roles in social psychology in which role theory has been systematically investigated. Roles seem to be phantoms. They are everywhere and every explanation seems reasonable, but it is difficult to grasp, identify, and completely specify them. Hence, there is still a need to clarify the role concepts in order to support the design of better information systems.

The paper is organized as follows. Section 2 clarifies the terminologies and aspects mainly concerned in this survey; Section 3 analyzes roles as modeling mechanisms; Section 4
discusses roles as modeling mechanisms specially in agent systems; Section 5 clarifies roles as management strategies; Section 6 illustrates the applications of roles in collaborative systems; Section 7 explains the role concepts in social psychology; Section 8 synthesizes the common ideas of roles applied in the different areas and gives an overview of roles; Section 9 discusses the challenges and benefits of role-based systems and Section 10 concludes this paper and indicates the further research topics.

2. TERMINOLOGY AND FOCI OF THIS SURVEY

Our motivation to survey the role concept and its applications is to obtain deeper insights into why and how roles are applied. We believe that there should be a common understanding of the same terminology, even though different contributors perceive it from different viewpoints. Consequently, we can better apply it as an underlying mechanism to system analysis, design and construction. We should understand it to avoid the experience of “the blinds for the elephant”. In this survey, we emphasize that roles can be taken as an underlying mechanism in system design and implementation. We hope to help designers and developers master and then apply roles to the development of their information systems.

We admit that even though we have cited many references, we must ignore quite many that mention roles, partially because they use “role” as a common word but not well-defined concept. However, we believe all the literature we cite together convey the basic ideas and applications of roles. In this survey, we use their motivations, contributions and conclusions to discuss their work on roles. To clarify and classify different role concepts, we arrange roles into five categories: object modeling, agent systems, Role-Based Access Control (RBAC), CSCW, and social psychology and management.

This survey uses the following terminologies: Data are plain data stored in a system possibly in different formats or data structures. Objects are composed of data and operations that access the data. Classes are used to describe objects’ common operations and data structures. Objects are instances of their classes. Objects are normally accessed. Agents are entities that possess the intelligent properties, such as, autonomy, social ability, reactivity and pro-activeness [90]. Users are human beings who use information systems. A group is a set of entities such as objects, agents, or human beings. Interfaces are the interaction methods between system components such as objects, agents, and software or hardware components. User interfaces are the interaction methods between users and systems. A system is an assemblage of parts forming a complex or unitary whole that serves a useful purpose [13]. We emphasize that a system is considered as a group of interdependent entities that interact regularly to perform a task. Here, entities can be expanded to express objects, agents, software or hardware components, software systems, hardware devices, human beings and groups.

After studying the contributions on roles in management, sociology and psychology, we conceive that the ideas of roles in social psychology cover what a role should be, because an information system is a virtual community that simulates a real one. To understand and compare different roles, we can observe them from the following different aspects:

By the reasons why roles are applied, we have:
- **Evolution**: roles are entities that express different states of objects at different stages of evolution.
- **Separation of concerns**: roles are entities that express different aspects of a role player at different context and time.
- **Interaction**: roles are used as tools to facilitate interactions among entities such as people, agents or objects.
- **Classification**: roles are used to classify entities to simplify the administration of the entities. Authorization is a special classification, i.e., roles are used to assign rights to entities to access system resources.

By role players, we have:
- **Operation**: the roles are played by an execution of a method of an object.
- **Object**: the roles are played by objects.
- **Agent**: the roles are played by agents.
- **Human**: the roles are played by humans.
- **Group**: the roles are played by groups of the above players.

By the contents roles should contain, we have:
- **Requests**: Roles are entities that facilitate role players to access system resources (files, objects, and devices) or requests for services. Roles are tags or tickets attached to role players to access objects or request servers to serve. In this category, rights of RBAC-roles (see Section 5) are evidently special forms of requests.
- **Services**: Roles are entities that facilitate role players to provide specific groups of services to the outside world. Responsibilities are in fact the interface part of services.

By whether roles have concrete specifications, we have:
- **Interface**: Roles are abstract entities to express the interfaces between role players. Roles only specify what the services (responsibilities) and requests (rights) are. How they are processed depends on their players. Here, we extend interface to a generalized term including user interface.
- **Process**: Roles are concrete descriptions in specifying the behaviors of their players. They specify not only what services (responsibilities) and requests (rights) are but also how they are processed.

By the meaning of role assignment, we have:
- **Instantiation**: to create an instance based on a role, i.e., a role is taken as a template to create a part of an object.
- **Allocation**: to attach a new entity to a player, i.e., a role is a different entity (with behaviors and structures) from the player.
- **Constraint**: to mandate constraints to a player, i.e., a role is a wrapper for the player to interact with outsiders.

By how roles are related with each other (role relationship), we have:
- **Conflict**: some roles cannot be played at the same time or in a period of time by one player.
- **Qualification**: to be qualified to play some roles, a player must have played some other roles at first.
- **Inheritance**: some roles may inherit other roles in a class hierarchy.
• Place: one role may be located among many other roles to serve some and request the others.

3. ROLES IN OBJECT MODELLING

To model objects in a problem domain, we require direct mapping and abstraction. Roles are taken as one kind of abstraction mechanism in modeling. We call these models as role-based models and the roles in them as modeling-roles.

In role-based models, a modeling-role is considered as an abstraction and decomposition mechanism related to objects. When an object plays a role, it accepts messages and provides services related to its role. A role constitutes a part of an object’s behavior that is obtained by considering only the interactions of that role and hiding all other interactions. The following characteristics of roles are commonly accepted in the literature of modeling methodologies:

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

The following characteristics of roles are not taken as commonly accepted ones:

1) An object and its roles have different identities [26] or share identity [4].
2) Roles can [49, 50] or cannot [82] play other roles.
3) Access to an object is [27, 51, 77] or is not [81] restricted by a role.
4) A role has meaning only in a role model [47] or has meaning generally [25, 49, 50].
5) Roles have [49, 50] or have not [81] their own states and behaviors.

In fact, some authors take roles as behavior describers [68]; some as objects possessing both data and operations [36, 77]; some as interfaces [83]; and some as external properties of objects [49, 50].

To better understand modeling-roles, we categorize the major reasons why roles are introduced as: (1) evolution of objects; (2) a fundamental modeling concept; (3) the interfaces among objects; and (4) the separation of concerns.

3.1 Evolution of Objects

To express the evolution of objects, roles are considered as states and tags. The major concern is how to organize the roles in a well-defined structure and associate them with objects.

Roles are first proposed to construct new types of databases [8] based on the role concepts carried by file records and relational file tuples. A role is a framework and has two meanings: commonalities and object evolutions. The framework may be applied by different kinds of objects at different times. Following this idea, Reimer designs a data structure to express a role, representing the earliest trial on supporting roles with a data structure [70]. Similar ideas are further developed to express evolutions of objects [4, 36].

Albano et al. implements an object-oriented programming language Fibonacci with roles [4]. Their role concept is similar to that from [8] but concentrates on the evolution of data objects. Roles are used to express the different states of an object. An object may change its identification to express the different roles it plays. The roles are designed as classes. An object is copied in part to form a new instance with the new role class. Therefore, the new instance and the original instance are combined to form the state of the object. The emphasis that an object has a set of roles is similar to the idea of Pernici [68]. We can call them as OODB-roles. Many subsequent contributions apply such a role concept. This idea is arguable in that objects are not born for special roles, but should collect abilities to serve and change their identities at first and then to play roles. With this concept, objects change identities. However, evolution generally means the same identity.

Gottlob, et al.’s role hierarchy [36] is aiming to support object evolution. Their roles are actually special classes that support specific object properties and services. Therefore, roles support specific behavior that is different from that of the relevant classes. Their work demonstrates that a class-based object-oriented system can be extended with roles and that role hierarchies can enhance basic object-oriented principles such as classification, object identity, specialization, polymorphism and behavioral contexts. Schreland and Thalhammer [77] continue the above research in Java and demonstrate that roles can also be expressed and applied in Java from the object evolution viewpoint.

3.2 Fundamental Modeling Concepts

The comprehensive studies on roles as fundamental concepts and mechanisms [16, 36, 39, 49, 50, 82] provide useful guidelines for role applications. In KL-ONE, a knowledge representation language [16], role is the primary component of knowledge representation. From an object-oriented point of view, a role acts like a method (or a two-place predicate) of a class (or a Concept), representing potential relationships between objects (or instances of different Concepts).

In the sense of ontology, Guarino discusses roles as the basic concept of KL-ONE-like languages [39] that support knowledge level modeling. The motivation of this work is to view roles as entities to express knowledge [16]. From this viewpoint, roles are defined as a concept that is founded but not semantically rigid. By “founded” it means that a concept can exist essentially independently and by “semantically rigid” it means that a concept contributes to the identity of its instances. The definition can help determine if a concept is a role. Roles allow not only for the representation of multiple views of the same phenomenon, but also for the representation of changes in time. Roles are also the bridge between different levels of details in an ontology structure and between different domains of networking ontology.

Kristensen investigates the role concepts in object systems [49, 50]. “A role of an object is a set of properties which are important for an object to behave in a certain way expected by a set of other objects [50].” It includes not only a set of methods but also states when it is instantiated. The work concludes that roles help the analysis, design and implementations of object systems.
Steimann [82] surveys a variety of the research and applications of role concepts in order to provide a consistent role concept and tries to overcome the drawbacks of the early contributions in his LODWICK modeling language. A role as a kind of protocol specifies the behavior and characteristics of an object. It can be viewed in different ways: a named relationship; a specialization or generalization; and adjacent instances. By adjacent instances, he means that an instance of a role class can be attached to an object to express that the object plays the role. From this point of view, roles are totally dependent on the object instances that play them and do not carry their own identity.

3.3 Interfaces among Objects

Roles as interfaces only specify what the role players (or objects) should do and the details of how to do are totally up to the players (or objects). This idea can protect system analysts and designers from concerning details of objects.

Genillord and Wegmann [35] think that a role is an abstraction of the behavior of an object. It consists of a subset of the interactions of that object together with a set of constraints on when they may occur. It always belongs to a specific larger behavior that involves other roles, called a collaborative behavior.

There are some different ideas on the relationships between roles and interfaces. Genillord and Wegmann’s definition stresses the difference between a role and an interface in object systems. Its purpose aims to describe collaboration among objects.

Steimann directly proposes and uses a formula “role = interface” [83] by analyzing the role concepts applied in object-oriented programming (OOP) and the Unified Modeling Language (UML) notational system. The problem to consider a role as an interface is that it considers only the responsibilities of an object but ignores the rights.

The collaborative behavior of a role represents the specific context in which it is defined, together with other roles in UML [81]. All the actions in collaborative behavior belong to one or more of its roles. In object modeling, roles are taken as dependent concepts of objects. In other words, roles cannot exist without objects. Every object may collaborate with other objects and should play a role in the collaboration.

3.4 Separation of Concerns

To express the separation of concerns of objects is another aspect for roles to consider. This idea can help “divide and conquer” and provide a new way for analysts and designers to divide and distribute tasks. The earliest work is done by Pernici [68]. Her work is motivated by the role concepts applied in office automation systems. An object can play different roles at different times, and may also play more than one role at the same time. Partitioning messages for an object relating to different roles has the advantage to allow designers to concentrate on the life cycle of an object in one role at a time. According to [68], a class is defined with roles as components:

- Class ::= {Nc, R0, R1, ..., Ra}
- Role ::= <Nc, P, S, M, Ra>

where Nc is an identification of a class, Nc is an identification of a role, P is a set of properties, S is a set of states, M is a set of messages, and Ra is a set of rules and R0,a are roles. This definition actually divides all the methods and properties of a class into different groups. Interestingly, the separation of concerns by roles had not made much advance since this work in 1990. New progresses are later made in aspect-oriented programming [47].

Pernici’s method is too complex to be practical as indicated in [68]. For example, one full page fails to describe a class Car with all the roles for a car. To describe a class, one must clearly specify all the roles that might be played by the objects of this class. It is almost impossible to predefine all the roles for a class of objects.

Among the implementations and applications of role concepts, Dahchour, et al.’s role model with meta-classes is worthy of mention [26]. After reviewing all the relevant literature, they define roles as the ones composed of responsibilities, facets, or aspects, and design their own role model. By their implementation, roles can be used to support the dynamic change of classes, multiple instantiation of the same class, and context-dependent access. Demsky and Rinard [27] and Kuneck et al. [51] propose a technique for program developers to analyze and understand the memory properties of objects. In their proposed technique, roles are a useful abstraction in helping developers explore and understand the behavior of object-oriented programs. They view roles as abstract object states intended to be of interest to the developers. The similarity between this work and the modeling roles is the separation of concerns. Their work mainly concerns about the separate states of objects in their running situations. In their system, roles are used to formalize the concept of a type that depends on the referencing relationships. Each role declaration provides complete aliasing information for each object playing that role. Different roles may have different implementations of methods. The role analysis technique [27, 51] is proposed to analyze dynamic programs by reflecting important aspects of roles: the separation of concerns and role transitions.

There are two major limitations in a modeling-role. One is that the role modelers stand among objects. When they model roles, they mainly consider the collaboration among objects. They do not consider the aspects of human users of the systems. Few contributions in modeling methodology inherit some good ideas from social-roles to be discussed later. The other is that roles are taken as a dependent concept of objects, i.e., objects are composed of roles. In fact, roles can be defined independently just the same as that classes can be defined before defining objects. That is to say, roles can be specified by analysts and designers in system analysis and design before objects are defined and created. Therefore, based on the traditional view of data modeling, it is difficult to obtain the whole view of a role in collaboration. However, in reality, when an object plays a specific role, it may have some special rights to access other objects. Considering another viewpoint when we design a class of objects, we need to know what services other classes of objects have provided. In modeling, roles should be identified and specified before designing classes. Roles are guidelines for modelers to design classes. It is therefore arguable for roles to have such properties as dependency: a role can not exist without the object; identity: the role and the object have the same identity; and locality: a role only has meaning in a role model [47].

Table 1 summarizes the major work that mainly discusses roles in role-based models and classifies them based on the issues they intend to address.
Table 1 Modeling-Roles

<table>
<thead>
<tr>
<th>Issues</th>
<th>Year</th>
<th>Authors</th>
<th>Motivations</th>
<th>Contributions</th>
<th>Conclusions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object evolution</td>
<td>1977</td>
<td>Bachman, Daya</td>
<td>To express the evolution of data objects</td>
<td>Propose to introduce roles in data models</td>
<td>The role concept can be introduced into data modeling.</td>
</tr>
<tr>
<td></td>
<td>1985</td>
<td>Reimer</td>
<td>To represent roles with a data structure</td>
<td>Propose a representation construct for roles</td>
<td>There should be a framework of roles relevant to the subtyping hierarchy.</td>
</tr>
<tr>
<td></td>
<td>1993</td>
<td>Albano, Bergamini, Ghelli, Orsini</td>
<td>To implement a database system that supports object evolution</td>
<td>Design and implement an object data model with roles applied, i.e., Fibonacci Database</td>
<td>Roles are a good mechanism to express object evolution. With their help, objects may acquire new types and behavior, retain their identity and preserve encapsulation.</td>
</tr>
<tr>
<td></td>
<td>1996</td>
<td>Gottlob, Shrefl, Rock</td>
<td>To support roles within a Smalltalk system</td>
<td>Extend Smalltalk to support roles with a role hierarchy</td>
<td>It is practical to introduce role mechanisms into an OOP language to express object evolution.</td>
</tr>
<tr>
<td>Fundamental concept establishment</td>
<td>1985</td>
<td>Brachman</td>
<td>To clarify such key concepts of KL-ONE as description, attribute, concept, role, inheritance, and instantiation</td>
<td>Introduce roles to represent knowledge in KL-ONE</td>
<td>In KL-ONE, roles are predicates to express the relationships between instances of Concepts.</td>
</tr>
<tr>
<td></td>
<td>1992</td>
<td>Guarino</td>
<td>To discuss ontology with clearer concept structures</td>
<td>Comprehensively discuss roles and concepts with ontology</td>
<td>Concepts can be categorized as roles, attributes, and natural concepts that have no overlapping with others.</td>
</tr>
<tr>
<td></td>
<td>1996</td>
<td>Kristensen</td>
<td>To improve object systems design</td>
<td>Deeply discuss roles at the conceptual level</td>
<td>Roles must be bound to objects.</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>Steimann</td>
<td>To make roles as the first class abstraction mechanism</td>
<td>Survey thoroughly the role representations and propose a formal role modeling language</td>
<td>The proposed models can help clarify the characteristics of roles and overcome the difficulties to integrate a clear role concept into the existing modeling frameworks.</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>Genilloud, Wegmann</td>
<td>To clarify the role concepts in a standard Reference Model of Open Distributed Processing</td>
<td>Discuss and present the differences between roles and other concepts of OOP</td>
<td>It is acceptable that roles are considered in different views.</td>
</tr>
<tr>
<td></td>
<td>2004</td>
<td>Dahchour, Pirotte, Zimanyi</td>
<td>To support the implementation of roles</td>
<td>Propose a role model to support role relationships between objects and roles with meta-classes</td>
<td>Metaclss is a better mechanism to support role concepts.</td>
</tr>
<tr>
<td>Interface design</td>
<td>2000</td>
<td>Steimann</td>
<td>To improve the dynamic and multiple classification of OOP</td>
<td>Analyze the interface of OOP and the role concepts in UML</td>
<td>Roles are interfaces among objects in an object system.</td>
</tr>
<tr>
<td>Separation of concern and rule transition</td>
<td>1990</td>
<td>Pernici</td>
<td>To design classes with roles</td>
<td>Propose a role concept to model object behavior</td>
<td>To completely describe a real-world object, roles are required.</td>
</tr>
<tr>
<td></td>
<td>2002</td>
<td>Demsky, Kuncak, Lam, Rinard</td>
<td>To analyze the dynamic memory occupied by objects in a running program</td>
<td>Propose a method for role analysis for objects in a data structure context</td>
<td>Analyzing objects’ behavior with roles could suggest the management policies for object memories.</td>
</tr>
</tbody>
</table>

3.5 Modeling-Roles in Specification and Design

In this category, roles are taken as concepts or tools to specify processes and designs in mapping problem domain to solution one. They are summarized in Table 2.

Holt pioneered in the research on role activity theory [45]. The theory was improved by Ould who proposed role activity diagrams (RAD) to describe software processes [64]. RAD is generally used to show the responsibilities, drivers and parallelism of the processes. A RAD comprises one or more role symbols annotated with role names. In RAD, a process is composed of roles. A role is composed of activities and taken as a means of associating human and other resources with tasks and processes. Murdoch and McDermid propose a method to model engineering design processes with RAD [56]. Various resources and events are associated with a role. A process involves the concurrent activity of several roles at one level. RADRunner [74] is built as a tool to facilitate RAD in expressing business processes. This activates a new wave to apply roles in the process of analysis and design of software. The roles of RAD are evidently process-roles. Roles are also used in order to encapsulate functionalities that may change dynamically when an object evolves. In UML,
associations represent relationships. An association has two ends. Roles are used to specify the ends. For example, an association between two groups of persons can be that a person as a manager manages many persons as employees. To provide appropriate mechanisms in UML for simulating roles, Steimann revises the UML's role concept in [81]. The diversity of concepts for UML to accommodate makes the UML diagrams difficult to understand including those expressed with role symbols as proposed. He clarifies that roles and relations are mutually dependent concepts, i.e., classes serve as the providers of instances and as takers of the responsibility for the realization of whatever the roles of a model promise; roles are partial specifications of classes; and roles give interfaces a prominent conceptual abstraction. His role concepts can replace the notions of association role and association end role as well as the rarely used association generalization.

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Motivations</th>
<th>Contributions</th>
<th>Conclusions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1983</td>
<td>Holt</td>
<td>To describe software process formally</td>
<td>Propose the role activity theory</td>
<td>Role activities can be introduced into modeling an engineering process</td>
</tr>
<tr>
<td>1995</td>
<td>Ould</td>
<td>To improve Holt's work by providing a notation system for RAD</td>
<td>Propose the notation systems for role activity diagrams</td>
<td>Roles are a good concept in notational tools to describe system processes formally</td>
</tr>
<tr>
<td>1996</td>
<td>VanHilst Notkin</td>
<td>To improve code reuse and adaptation with roles</td>
<td>Propose a method to map role-based design to C++ implementation</td>
<td>Role-based design can be mapped to C++ templates</td>
</tr>
<tr>
<td>1998</td>
<td>Riehle Gross</td>
<td>To use role modeling as an enabling technology for framework design</td>
<td>Propose a role model-based framework for object-oriented software development</td>
<td>Using role models to design frameworks to specify the interaction between objects and between the frameworks and their clients</td>
</tr>
<tr>
<td>1999</td>
<td>Kendall</td>
<td>To implement role mechanisms</td>
<td>Propose a method to support role models with aspect-oriented programming</td>
<td>Aspect-oriented programming can support the implementation of the role model framework in [72]</td>
</tr>
<tr>
<td>2000</td>
<td>Murdoch</td>
<td>To apply RAD into engineering process specification</td>
<td>Propose a method to model engineering design processes with RAD</td>
<td>RAD can be used to model engineering processes</td>
</tr>
<tr>
<td></td>
<td>Steimann</td>
<td>To clarify the role concepts in UML</td>
<td>Propose a revised UML metamodel based on a much simpler role definition</td>
<td>The metamodel introduces the interface concept while avoiding the inconsistence problems related to the roles in UML</td>
</tr>
<tr>
<td></td>
<td>Zhao Kendall</td>
<td>To introduce roles to improve component design</td>
<td>Enable to identify components based on roles</td>
<td>Roles can be considered as abstraction and representation of collaborative components</td>
</tr>
</tbody>
</table>

Reenskaug et al. apply roles to describe object-oriented software engineering processes [69]. A role in an object specification is called an object type that is a specification of a set of objects with identical externally visible properties. A role is a "why" abstraction. All objects that serve the same purpose in a structure of collaborating objects in a certain context are said to play the same role. They emphasize the role model but not the role itself and introduce roles intuitively. It shows a good application of modeling-roles in describing engineering processes.

VanHilst and Notkin introduce roles with both object collaboration and evolution in designing reusable components. In the collaboration view, a role is the part of an object that fulfills its responsibilities in collaboration [88]. Compared to classes, roles encapsulate fewer decisions and are thus more stable with respect to evolution. They provide a method to implement roles with C++ class templates and add roles into a class to refine or extend its interface. The composition and inheritance are used to express the collaboration between roles in their implementation. Their work again demonstrates that a class-based programming language can be expanded to support roles with evolution requirement.

Riehle et al. introduce a role type to describe the view one object holds on another [72]. A role type is described by using a type specification mechanism. A role type is described by using a type specification mechanism. A role instance (they call them as role objects) is one that represents one specific role of an object (they call this kind of object as core object to be differentiated from role objects) to its clients. The role instance wraps the object. The object maintains its role instances for different clients. Collaboration is taken as a set of roles and their relationships. Hence, the design of a framework can be composed of role models, each of which may be a pattern instance. A role model actually describes a particular aspect of an object. Their roles support the separation of concerns. It is clear that this work improves Reenskaug's [69] by providing the modeling roles that support framework design and integration.

Kendall discusses a methodology to design role models with aspect-oriented programming [47]. She adopts and implements the role concepts of modeling-roles [49, 50]. According to her, roles are abstraction and decomposition mechanisms. A role can be specified in language AspectJ.

Bäumer et al. describe context-specific views of an object with role-object pattern as separate role objects [10]. An object class is separated into two subclasses: core and role. Role objects are dynamically attached to and removed from the core object. They use subclasses of the role classes to express the roles played by the instances of the object class. Their method provides a good guidance to design an object system with role concepts. It helps the role concepts be reused at the pattern level.
Zhao and Kendall propose the applications of role modeling in component design [93]. A role model is stated as an abstraction that describes the patterns of interactions among a set of entities. The entities play certain roles in a given context. The context is captured by the role model. A role model depicts frequently occurring but transient relationships among entities or objects that are working together to perform a certain task or accomplish a certain goal.

4. ROLES IN AGENT SYSTEMS

Many agent systems apply different aspects of the role mechanisms. The role concepts are generally used in agent systems to describe the collaboration among cooperative processes or agents. We call these roles in agent systems as agent-roles. Because agent concepts evolve from objects [58, Page 1, Fig.1], agent-roles inherit the properties of modeling-roles.

Becht et al. pioneered the concept of considering roles as interfaces and developed a multi-agent system ROPE [11]. The cooperative processes are described with roles. An agent that wants to take part in the cooperation must fulfill the service requirement specified by a role. In their system, a role is formally defined as an entity consisting of a set of required permissions, a set of granted permissions, a directed graph of service invocations, and a state visible to the runtime environment but not to other agents.

Taking roles as specific processes is the main idea of Stone and Veloso’s soccer team formation [84]. A role consists of a specification of an agent’s internal and external behavior. They use roles to obtain a flexible team agent structure and a method for inter-agent communication. First, the team agent structure allows agents to capture and reason about team agreements. They express collaboration between agents by formations that decompose the task space into a set of roles. In their solution, agents can flexibly switch roles within formations, and agents can change formations dynamically, according to predefined triggers to be evaluated at run-time. This flexibility increases the performance of the overall team. Their teamwork structure further includes pre-planning for frequently occurring situations.

In agent-oriented software engineering, Gaia methodology [91, 98] is proposed to support system analysis and design by taking a multi-agent system as an organization. Analysis and design are well-separated phases, i.e., analysis aims to develop an understanding of the system structures by roles models and interaction models while the design phase aims to define the actual details of the agents in the system. An important contribution of Gaia is that it overcomes the weakness of roles in modeling-roles in that roles accommodate rights or permissions for agents to possess. In Gaia, roles are described with responsibilities, permissions, interaction protocols and activities. The ideas applying roles in agent-oriented software engineering are similar to those in our previous work [95, 97].

In Organization-Centered Multi-Agent Systems (OCMAS) [30], roles are emphasized as an important element of organizations. A role is the abstract representation of a functional position of an agent in a group. It helps overcome the drawbacks of agent-centered multi-agent systems, i.e., an agent may communicate with any other agents; an agent provides services available to every other agent; there are no constraints for agents to access others; and agents interact with each other directly. The roles in OCMAS are similar to those in [91]. Group roles are mentioned in [30] to mean the roles in a group, i.e., an agent must play a role in a group. This definitely incurs an argument of which should be based on the other: roles or groups [96].

Cabri et al. [18-20] apply both aspects of agent-roles: processes and interfaces. A role imposes a defined behavior to the objects playing it, thereby reflecting the process aspect. It also allows a set of capabilities, which can be exploited by agents to carry out their tasks, reflecting the interface aspect. This allows a role to serve as an abstract description for the functions an agent is responsible to fulfill in order to reach an assigned goal. From the agent-oriented approaches, roles are a proper means of refining agent-oriented models. Their work [20] considers rights as part of their agent-roles.

Based on the work of such psychologists as Biddle and Thomas [13] and Shakespeare’s role concepts in theaters, Odell et al. [59, 60] propose with both the aspects mentioned above a superstructure specification that defines the user-level constructs required to model agents, their roles and their groups. Their modeling constructs provide the basic foundational elements required in multi-agent systems to foster dynamic group formation and operation. Roles specify normative behavioral repertoires for agents and provide both the building blocks for social agents and the requirements by which agents interact. Individual organizations can thus track and control their behaviors by applying the role concepts to facilitate dynamic, controlled, and task-oriented group formation. The idea that some roles are group roles [60] is similar to those of [30], i.e., group roles are only meaningful in a group [30].

Partsaoulakis and Vouros review role concepts and mechanisms in multi-agent systems [66]. Roles are viewed as tools to manage the complexity of tasks and environments. High degree of interactions, environment dynamics and distributivity are emphasized to show the importance of roles. They use five aspects to analyze the past research on agent-roles: specification, dynamics of assignments, dynamics of roles, cardinality of roles and life-span of roles. From their analysis, most roles in this area are used both as 1) an intuitive concept in order to analyze multi-agent systems and model inter-agent social activity; and 2) a formal structure in order to implement coherent and robust teams.

In many contributions, both aspects of roles: interfaces and processes [18-20, 59, 60] have been discussed. Interface-roles are used in system analysis and design when architecture and structures are more important while process-roles are used in system implementation when concrete jobs should be specified clearly and exactly. Process-roles specify how the role players to take the responsibilities and how to use the rights.

Roles can be abstract as interfaces and concrete as processes. Roles have many aspects to consider in different usages. That is why roles can be applied into every stage in system development: analysis, design, and implementation. However, most authors fail to clarify when and how to separate them. It is an important point in agent systems because this ambiguity may bring in difficulties of system design and maintenance. Gaia [91, 98] makes this clear, i.e., roles models are composed at the analysis phase while agent models are composed at the design phase. From the viewpoint of systems engineering, interface-roles should be used in the system design to specify the relationships among system components while process-roles in system construction to implement components. This idea is also discussed in our previous work [95, 97].
In agent systems, the following consensus is reached [11, 18-20, 59, 60, 66, 84]:

1) A role can be taken as an interface.
2) A role can be taken as a process.
3) A role instance is deleted when an agent is destroyed, i.e., its lifetime depends on its agents.
4) 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.
5) Roles have three functions: comprise special behavior, form the behavior of an agent, and set a place for an agent in a group (or define the inter-relationships among agents).
6) Roles can be used for expressing the organizational structure of a multi-agent system.
7) Roles can be used for specifying interactions in a generic way.
8) Roles can be used as agent-building blocks in class diagrams.

Table 3 presents the general picture of the major work on agent-roles.

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Motivations</th>
<th>Contributions</th>
<th>Conclusions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999</td>
<td>Becht</td>
<td>To increase the adaptability of the structural change of an agent system</td>
<td>Propose an architecture ROPE to support roles in agent systems</td>
<td>Roles can be used to support the co-operative processes among agents.</td>
</tr>
<tr>
<td>1999</td>
<td>Stone Veloso</td>
<td>To provide a well-defined team structure</td>
<td>Propose a teamwork structure with roles</td>
<td>Agents can respond to changing environments by dynamically changing their roles.</td>
</tr>
<tr>
<td>2000</td>
<td>Wooldridge Jennings Kinny Omicini Zambonelli</td>
<td>To introduce organizational methods into agent-oriented software engineering</td>
<td>Propose a new coordination model and a new method Gaia to support agent-oriented software engineering</td>
<td>Roles can be used in the analysis phase of agent-oriented software engineering and agents in the design phase.</td>
</tr>
<tr>
<td>2001</td>
<td>Depke Heckel Kuster</td>
<td>To analyse the application of roles in agent systems</td>
<td>Analyze the role requirement and propose a method to transform UML diagrams into implementation with the help of roles</td>
<td>Roles can be used in many ways in modeling agent systems, such as expressing an organizational structure, specifying interactions, and serving as agent-building blocks.</td>
</tr>
<tr>
<td>2003</td>
<td>Odell Van Dyke Parunak Fleischer</td>
<td>To simplify the design of sophisticated agent systems</td>
<td>Clarify and argue that the roles for agent systems should be consistent with those in management</td>
<td>Roles are good implications of how agents behave in a group.</td>
</tr>
<tr>
<td>2004</td>
<td>Partsakoulakis Vouros</td>
<td>To support the collaboration among agents</td>
<td>Emphasize the importance of roles for agent systems by analyzing their properties in agent systems</td>
<td>The extensive use of roles in implemented systems shows the need for role-oriented thinking and modeling in agent system development.</td>
</tr>
<tr>
<td></td>
<td>Cabri Ferrari Leonardi</td>
<td>To investigate how the concept of role can be exploited in agent systems and how to simplify the related tasks</td>
<td>Propose and present the framework BRAIN that exploits the concept of role in different phases of the development based on a simple yet general role-based model for interactions</td>
<td>The use of roles can bring different advantages, in terms of separation of concerns between algorithmic issues and interaction issues, generality of approaches, locality, and reuse of solutions and experiences.</td>
</tr>
<tr>
<td></td>
<td>Ferber</td>
<td>To overcome the drawbacks of agent-centered multi-agent systems</td>
<td>Propose organization-centered multi-agent systems by introducing roles</td>
<td>The OCMAS with roles overcomes the drawbacks of agent-centered multi-agent systems.</td>
</tr>
<tr>
<td>2005</td>
<td>Cabri Ferrari Leonardi</td>
<td>To support agent interactions and increase agents’ adaptibilities</td>
<td>Implement a role interaction infrastructure that enables Java agents to dynamically assume and use roles at runtime</td>
<td>Roles can be more useful to design, develop, and even maintain complex applications, where there are many interactions among interacting agents.</td>
</tr>
<tr>
<td></td>
<td>Odell Nodine Levy</td>
<td>To design a metamodel for agents and groups and facilitate agent system development</td>
<td>Propose a metamodel for agents, roles and groups</td>
<td>The metamodel enhances the predictability, reliability and stability of agent systems.</td>
</tr>
</tbody>
</table>
5. ROLE-BASED ACCESS CONTROL

5.1 Evolution of RBAC-Roles
Owing to numerous publications on RBAC [2, 3, 31-32, 76] in recent years, we cannot cover them completely. We can discuss only those contributions mainly concentrating on role mechanisms. We call the roles applied in RBAC as RBAC-roles. They are proposed as one way to deal with the accessibilities of users to numerous objects or resources in a system. The central question is to construct an efficient data structure for RBAC-roles in order to express a user’s right to access them.

Ferraiolo and Kuhn [31] pioneered in the research of RBAC. It is well accepted that designing security for situations with many users and objects is greatly facilitated by using a role-based design. RBAC aims to apply roles to simplify the tasks of security administrators in order to enforce access control policies. A role is described as a set of transactions that a user or set of users can perform within the context of an organization. Since then, RBAC has developed rapidly and received much attention in computer security and protection. Many discuss the architectures and mechanisms of RBAC [2, 3, 31-32, 76]. Their research demonstrates that roles are excellent underlying mechanisms in dealing with access control and system security. However, only a few discuss the role concept even though RBAC is designed based on it.

Sandhu et al. [76] state that a role can represent specific task competency and embody the authority and responsibility of a user. Roles define both the specific individuals allowed to access resources and the extent to which resources are accessed. In [3], a role is defined as a set of actions and responsibilities associated with a particular activity. The persistence of roles in organizations is stressed. This property helps RBAC provide a powerful mechanism for reducing the complexity, cost, and potential for error in assigning permissions to users within an organization.

The Generalized RBAC (GRBAC) model extends traditional RBAC models [22, 55] by incorporating the notion of object roles and environment roles, with the traditional notion of subject roles. From the viewpoint of security, subjects are entities to access objects; objects are entities to be accessed; and environments are entities that help subjects access objects. Object roles capture various commonalities among the objects and emphasize the classification of objects. Environment roles capture security-relevant information about the environment and are used to express common states for the objects and subjects to behave. The Temporal-RBAC (TRBAC) model supports periodic role enabling and disabling and temporal dependencies among permissions [12]. It introduces time into the access control infrastructure. The Generalized TRBAC (GTRBAC) model captures an exhaustive set of temporal constraint needs for access control [46]. It expresses periodic as well as duration constraints on roles, user-role assignments, and role-permission assignments. In an interval, the activation of a role can further be restricted as a result of numerous activation constraints including cardinality constraints and maximum active duration constraints. The GTRBAC model extends the syntactic structure of the TRBAC model. Its event and trigger expressions subsume those of TRBAC. Furthermore, it allows expressing role hierarchies and separation of duty constraints for specifying fine-grained temporal semantics.

When discussing a proposed standard for RBAC, Ferraiolo et al. [33] propose a role as a means for naming many-to-many relationships among individual users and permissions. A role is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role. The role characteristics in this category are [12, 22, 31-33, 46, 57, 76]:

- 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.

Their work makes RBAC-roles have a mature, consistent and standardized definition. The role concept in RBAC actually comes from the idea used in operating systems. A role is a tag that can be used by a system to perform protection on resources in it. The system grants users based on their roles the access rights to files in order to accomplish protection. The major problem is that it considers only the benefits obtained by assigning access permissions to users. This is a common view of software systems, i.e., a user is a client and a system is a server. RBAC-roles only emphasize roles in the aspect of rights. Compared with the modeling-roles, RBAC-roles are practical and sufficient in many applications.

On the other hand, RBAC takes roles as specific static identities to differentiate among users. That is to say, a role is stated as a group of users to mean that if administrators can assign accessing rights to a role, they actually assign the rights to a group of users who play this role. This idea is to design some security strategies based on definite operations on resources. In general, a role-based information system requires that a role be dynamic. In other words, a user may bind to different roles from application to application and possibly during a single session within an application. Clearly, this is not captured by RBAC-roles.

RBAC has been researched actively for several years. A standard for RBAC is proposed [33] and gradually applied into administrative work and modeling [23]. The basic ideas from RBAC are now applied to database management such as Oracle [62], SQL server [21] and Massachusetts Institute of Technology (MIT)’s Roles Database [71]. Although there are some trials on the research of fundamental and generalized role models [57], the basic idea is still restricted to the administrative requirement. The basic view for roles in RBAC is now well-accepted, i.e., roles express the rights for users to access system objects. To generalize the role concepts of RBAC, many new contributions introduce the ideas of modeling roles, object technologies and temporal data models into RBAC [12, 22, 46] to widen the applications.
5.2 Applications of RBAC-Roles

In most practical database systems, roles are taken as a tool that allows an administrator to collect users into a single unit against which one can apply permissions. This means that the mechanism provided by RBAC is successfully used in RDB (relational database) systems. We consider these to be RDB-roles.

MIT implemented a system, called the Roles Database, to centrally manage people’s authorization for computer-based enterprise-wide applications [71]. This system supports an environment where many people are authorized to perform similar tasks, but for different departments or fiscal areas.

In the Roles Database, a role is an authorization including a 3-part entity consisting of a Person, Business Function, and Qualifier.

1) A person is represented by a username within mit.edu domain.
2) A function is one of the tasks that a user can be authorized to perform in a given system. Functions are grouped by applications. Each function is associated with a specific qualifier type. When an authorization is created, the qualifier chosen must be of the right type to match the function.
3) A qualifier can be an account number, organization number, budget group, etc. Since the qualifications of each type are organized into a hierarchy, a qualifier can also be a branch of the tree of account numbers, a branch of the tree of organizations, etc.

Roles or authorizations are centrally defined in understandable business terms, and then converted to the native representation of each application to which they apply.

Oracle 9i introduces roles and privileges [62]. User privilege is a right to execute a particular type of SQL statement, or to access another user’s object. Roles are created by users (usually administrators) and used to group together privileges or other roles. They are a means of facilitating in granting multiple privileges or roles to users.

In Microsoft SQL Server 7.0, roles are also introduced as a tool that allows one to collect users into a single unit against which one can apply permissions [21]. Therefore, roles are taken as a group of users. Permissions granted to, denied to, or revoked from a role also apply to any members of the role. One can establish a role that represents a job performed by a class of workers in one’s organization and grant the appropriate permissions to that role. As workers rotate into the job, one simply adds them as a member of the role; as they rotate out of the job, remove them from the role. One does not have to repeatedly grant, deny, and revoke permissions to or from each person as they accept or leave the job.

The permissions are applied automatically when the users become members of the role. Basically, their roles are RBAC-roles.

Lupu and Sloman propose a role-based framework for distributed system management [54]. Although they mention some role mechanisms applied in different areas such as management, Computer-Supported Collaborative Work (CSCW), object modeling, and process specification, their role template specification is similar to those in RBAC, because their aim is to provide a framework for system management. They take a role as a set of policies relating to a particular manager position and the notation to specify policies. A role identifies the authority, obligations, functions and interactions associated with a position. They notice that a role framework has to be very flexible to accommodate peer-to-peer as well as hierarchical relationships and to permit multi-party interactions. Roles are a good tool to form such kind of frameworks.

To meet more administration requirements, ARBAC97 (administrative RBAC’97) is proposed in [75] to deal with role-based administration tasks such as user-role assignment, permission-role assignment, and role-role assignment while RBAC is more concerned about authorization. Moreover, ARBAC02 [61] is proposed to improve ARBAC97 by removing unnecessary integration of the user and permission pools and the role hierarchy. In ARBAC02, the organization structure is introduced as pools of users and permissions and the pools are independent of role or role hierarchy. This direction is to surpass the limitation of RBAC and apply RBAC-roles in system administration. We can call the above roles [54, 61, 75] as system-roles. The system-roles consider more on such external properties as relationships among roles, users, and permissions.

Table 4 summarizes the major work on roles in RBAC.

6. ROLES IN CSCW SYSTEMS

People are more and more involved in Computer-Supported Cooperative Work (CSCW) because of the pressure from companies to improve their product-development and decision-making processes, and the convenience brought by the information super-highway [92]. CSCW systems are computer-based tools that support collaborative activities and should meet the requirements of normal collaboration. They should not only support real/virtual face-to-face collaborative environment [85] but also improve face-to-face collaboration by providing more mechanisms to overcome its drawbacks of face-to-face collaboration, for example, an aggressive person may manipulate a meeting. It is natural for CSCW systems to apply the role concepts [94]. We call such roles as CSCW-roles.

Even though roles were introduced by Leland et al. into CSCW in 1988, the research on roles as underlying mechanisms of such systems has remained under-exploited. Some systems consider roles as labels for users [15, 38, 40, 87, 97]. Based on these labels, the system designers use switch/case-like structures to define different working processes for different roles. In other words, systems use an enumeration list to express roles and give each role a specific function to perform. After they are released, the roles may not be changed [97]. Some applications are built to support roles without clearly stating what roles are [38, 80]. Their roles are what people think intuitively. Others use the role mechanism similar to that of RBAC, i.e., restricting the access rights to the shared resources in the collaboration [29, 80]. They can only support pre-defined or static roles. Users have no support facility to tune their roles in order to make collaboration more productive and efficient. Some systems, e.g., the Coordinator [34, 89], are
criticized as “naziware”, because they over-restrict users. In other words, a user’s operations are over-restricted by the roles they are playing. This criticism motivates the research and application of dynamic roles [29, 80].

In EIES (Electronic Information Exchange System) [87], roles are built out of a subset of the primitive privileges (such as append, link, assign and use) that are crucial to a human communication process. In Quilt [48], there are roles for writers (who are allowed to change their own work only), readers (who are not allowed to modify the document), and commentators (who can only add “margin notes” to it). Such CSCW systems have lost flexibility after introducing roles because they can provide only static role mechanisms. They lack flexible mechanisms for role specification, tuning, change and transition because a mechanism to express a role is simply missing.

### Table 4 RBAC-Roles

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Motivations</th>
<th>Contributions</th>
<th>Conclusions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1991</td>
<td>Ferraiolo Kuhn</td>
<td>To simplify the tasks of system administrators and group users into roles</td>
<td>Initiate the research on RBAC</td>
<td>Roles are good mechanisms to offer protection and facilitate administrators’ jobs.</td>
</tr>
<tr>
<td>1995</td>
<td>Ferraiolo Cugini Kuhn</td>
<td>To clarify the role concept in RBAC</td>
<td>Propose a formal RBAC model, architecture, prototype, and demonstrations to validate the use and feasibility of RBAC</td>
<td>RBAC is adaptable to any organizational structure and means of conducting business. It provides greater productivity on the part of security administrators, resulting in fewer errors and higher operational security.</td>
</tr>
<tr>
<td>1996</td>
<td>Sandhu Coyne Feinstein Youman</td>
<td>To clarify the need, framework and applications of RBAC reference models</td>
<td>Propose and describe formally the reference models of RBAC</td>
<td>The proposed RBAC models provide a common framework for related research and development. There are still many problems to be investigated in RBAC.</td>
</tr>
<tr>
<td>1997</td>
<td>Lupu Sloman</td>
<td>To facilitate the management of systems</td>
<td>Propose a role-based framework for distributed system management</td>
<td>Roles are useful in helping managers administrate distributed systems.</td>
</tr>
<tr>
<td>1999</td>
<td>Sandhu Bhamidipati Munawer</td>
<td>To extend RBAC to manage user-role, permission-role, and role-role assignment</td>
<td>Propose a new role-based administration model ARBAC97</td>
<td>ARBAC97 is a formal role-based model that enforces more sophisticated administrative policies than RBAC does.</td>
</tr>
<tr>
<td>1999</td>
<td>Nyanchama Osborn</td>
<td>To clarify and solve the problem of role conflicts in the role relationships</td>
<td>Propose and describe a reference model for RBAC and role-graph model with its accompanying algorithms implementing role-role relationships</td>
<td>The proposed role graph, model and relevant algorithms provide a way of showing and solving the consequences of conflict of interest considerations in RBAC.</td>
</tr>
<tr>
<td>2000</td>
<td>Ahn Sandhu</td>
<td>To specify constraints that are an important aspect in RBAC</td>
<td>Propose a formal language RCL 2000 to specify constraints to avoid the ambiguities of natural languages</td>
<td>The persistence property of roles provides a mechanism for reducing the complexity, cost, and potential for error in assigning permissions to users within an organization.</td>
</tr>
<tr>
<td>2000/2001</td>
<td>Covington Moyer Ahmed</td>
<td>To unify ideas from several existing access control models into one model that captures all security-relevant state in a system.</td>
<td>Propose a generalized RBAC model GRBAC to support time-dependent access control</td>
<td>In security systems, roles can be categorized into subject-, environment- and object-roles. GRBAC extends the traditional RBAC model to support time-dependent access control.</td>
</tr>
<tr>
<td>2001</td>
<td>Ferraiolo Sandhu Gavrila Kuhn</td>
<td>To standardize the application of RBAC models</td>
<td>Propose a standard for RBAC</td>
<td>RBAC has found wide applications in computer system security administration. A standard is required to offer relevant mechanisms.</td>
</tr>
<tr>
<td>2001</td>
<td>Bertino Bonatti Ferrari</td>
<td>To support specification of temporal constraints</td>
<td>Propose the Temporal-RBAC (TRBAC) model that addresses some of the temporal issues related to RBAC</td>
<td>TRBAC supports for periodic enabling/disabling of roles, individual exceptions, and the possibility of specifying temporal dependencies among actions.</td>
</tr>
<tr>
<td>2002</td>
<td>Oh Sandhu</td>
<td>To overcome the weakness caused by top-down role hierarchy</td>
<td>Improve role-based administration model ARBAC97 to a new model ARBAC02</td>
<td>The organization structure can be taken as pools of users and permissions and the pools are independent of role or role hierarchy. The bottom-up method of ARBAC02 is better than the top-down one of ARBAC97.</td>
</tr>
<tr>
<td>2003</td>
<td>Crampton Loizou</td>
<td>To improve the existing RBAC models and develop a new role-based</td>
<td>Propose a role-based administrative model SARBAC that reduces inheritance in the role</td>
<td>SARBAC is complete, widely applicable, and versatile and offers significant practical and theoretical advantages over the past</td>
</tr>
</tbody>
</table>
Patterson [67] emphasizes a role concept with the idea of an interface between objects even though it is not clear how to support roles as admitted in [67]. A user-interface is simply an elaborate mechanism for moving messages back and forth between a user and objects. A role is an abstract user type. It is an object class of which a user is an instance. Given the roles of users, the messages understood by them are known. In the RENDEZVOUS system [67], the role is used to enable and disable an object’s visibility and to act as a filter on the input events. Patterson’s work implicitly expresses the access rights of users by roles. Unfortunately, the work [67] does not go far enough to provide a real abstract structure to support more flexible user interfaces based on roles.

Greenburg proposes to accommodate individual roles and group differences with roles in the SHARE system [38]. Personalized groupware can lead to the wider acceptance of a product by differences with roles in the SHARE system [38]. Personalized based on roles.

Edwards introduces access control policies and roles to avoid chaos in collaborative applications [29]. A role is described as a category of users within the user population of a given application; and all users of a certain role inherit a set of access control rights to objects within the application. By describing role membership, users can be relieved of some of the burden of tracking, updating, and anticipating role membership explicitly. Dynamic roles are supported by binding a predicate function to a symbolic role name and mapping from roles to policies. In fact, the essential characteristics of roles are determined by policies in this way. A role’s dynamic property can change by mapping a role’s name to a policy [29, 52]. This mapping is accomplished by associating a predicate function with them. A formal language INTERNEZZO [29] is developed to specify roles with attributes, predicates and policies. Dynamic roles are very important in collaboration.

The question arises: should a method be developed going further in this direction? Or, how can a policy be tuned, changed or transferred? Edwards actually applies into the policies the access control methodology that is to rely on an administrator to use the INTERNEZZO language to tune, change, or restrict access control policies. Hence, it is still difficult for end-users to do such tasks.

Smith et al. built the Kansas system in 1998 and emphasized the importance of roles [80]. People in a group play various roles even though they may not be well defined. Kansas intends to support roles by special treatments in multi-user interfaces. Roles are in general supported by a system’s treatment of output and user inputs, which are similar to the view of incoming messages and outgoing messages at a more abstract level. Thus, Kansas can support flexible roles by managing the accessibilities to the resources such as inputs and outputs. In fact, it applies part of the RBAC-role implicitly because their accessibilities are similar to the permissions of RBAC-roles. “The physics underlying reality does not define roles, though it supports them [80].” This partially answers why there are so many different aspects to consider in using them in information systems. Guzdiol et al. inherit the idea to deal with roles in [80] and develop many roles that are typical in their collaborative tool CoWeb [35] to accommodate a variety of users:

- They describe roles as specific concerns and activities associated with individuals who choose to use CoWeb; and
- They define a role as a human construct created and sustained by the interaction of minds. Their roles can be extended on and off-line and are not associated with capabilities within the shared space.

In fact, this work is another important practice similar to what Leland et al. tried in 1988 [52], i.e., roles are pre-designed and static.

Recently, Zhu and Zhou [94, 96] reviewed completely the role mechanisms applied in collaborative systems and proposed a new role definition. It is composed of both responsibilities and rights. Their work establishes the requirements for role-based collaboration; present the concept, requirements, and principles of role-based collaboration; propose a model E-CARGO for role-based collaboration; and describe the kernel mechanisms and their implementation to facilitate the development of role-based collaborative systems for industrial applications. The conclusion is that roles in collaborative systems are important and role-based collaboration requires more comprehensive research in order to achieve the mentioned goal to support and improve face-to-face collaboration. The role mechanism in E-CARGO extends the idea of Gaia in that roles specify requests which contain any kinds of outgoing messages while Gaia’s rights contain only specific access operations such as read, write and modification [94, 96].

In the SPADE-I [8] environment, Bandinelli et al. introduce static roles such as project manager, system administrator, designer, and programmer. Users can perform operations on the global workspace according to their roles. Because their major task is to provide an environment for software engineering, their roles are intuitive and simple. They just use a string such as “ProjManager” to express the role of a project manager. In other words, their roles are used to specify the tasks or operations of a user, i.e., CSCW-roles.

Consequently, Zhu et al. [97] apply their E-CARGO model to build a software development tool. Considering software development as a collaborative activity, they discuss the importance of roles in software engineering and that of role-based software engineering; propose and describe a role-based software process; and implement a prototype tool for developing complex information systems with the help of role mechanisms.

In summary, in CSCW there is neither commonly-accepted concept of roles nor methods to express a role. Some use roles as a commonly understandable word [40, 52, 87]; some support role ideas with some special user interface setting mechanisms [38, 80]; and some propose a tool to specify roles dynamically [29, 84]. However, 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. Table 5 offers an overview of the work on roles as interaction media in collaborative systems.

7. ROLES IN SOCIAL PSYCHOLOGY AND MANAGEMENT

After investigating the role concepts and mechanisms in different research and engineering areas in information systems, let us consider the role concepts from social psychology. Among the heavy literature, we can select only those most significant ones that discuss roles systematically and are highly relevant to information systems.

From Oxford English Dictionary [64], we can find the common illustrations about “role”. A role is

1) “The part or character which one has to play, undertakes, or assumes”;
2) “The part played by a person in society or life”;
3) “The typical or characteristic function performed by someone or something”.

The term “role” derives from the theater and refers to the part played by an actor. Also in [64], a role is defined in a behavioral and psychological view. It is “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.” We call such defined roles social-roles.

Table 5 CSCW-Roles

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Motivations</th>
<th>Contributions</th>
<th>Conclusions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>Leland Fish Kraut</td>
<td>To protect interactions among users</td>
<td>Build the earliest CSCW application Quilt with roles for users to play</td>
<td>Using roles to confine a user’s permission to access a shared document is practical.</td>
</tr>
<tr>
<td>1991</td>
<td>Patterson</td>
<td>To implement a multiuser interface</td>
<td>Propose roles as user-interfaces</td>
<td>Most multi-user applications require an ability to describe users in terms of roles.</td>
</tr>
<tr>
<td>1996</td>
<td>Edwards</td>
<td>To control the accessibilities to shared resources</td>
<td>Propose and implement SHARE that can express the personal interfaces and differences among groups with roles</td>
<td>Personalizable groupware can lead to the wider acceptance of a product by offering a system that conforms to individual needs.</td>
</tr>
<tr>
<td>1998</td>
<td>Bandinelli Di Nitto Fuggetta</td>
<td>To support cooperation among the team members in software development</td>
<td>Introduce static roles in an environment to support cooperative software engineering processes</td>
<td>Access control in collaborative environment can be dynamically adjusted by policies and roles.</td>
</tr>
<tr>
<td>2000</td>
<td>Smith Hixon Horan</td>
<td>To facilitate role requirements with shared spaces</td>
<td>Implement a collaborative system Kansas that supports role assignment to human users</td>
<td>It is possible to support roles in Kansas without defining them.</td>
</tr>
<tr>
<td>2006</td>
<td>Guzdial Rick Kerimbaev</td>
<td>To find an underlying role mechanism</td>
<td>Propose a new view on roles and point out the principles of roles applied in CSCW systems</td>
<td>Roles should become an underlying mechanism to build CSCW systems.</td>
</tr>
<tr>
<td>2006</td>
<td>Zhu Zhou</td>
<td>To build the foundations of role-based collaboration</td>
<td>Review the role mechanisms applied in collaborative systems and propose a new model for role-based systems</td>
<td>Role mechanisms are important in building collaborative systems and require more comprehensive research.</td>
</tr>
<tr>
<td>2006</td>
<td>Zhu Zhou Seguin</td>
<td>To build a role-based software development tool</td>
<td>Build a prototype of a role-based software development tool and emphasize that software development is a collaborative activity</td>
<td>People playing roles can interact with others at a definite time. In software development, developers playing roles can concentrate on their special tasks without being distracted by irrelevant affairs.</td>
</tr>
</tbody>
</table>

The major concerns of social-roles are how to avoid conflicts and ambiguities among the people related to a position [7, 13]. The literature mainly discusses human resource management and staff recruitment. Thus the basic question turns into how to present an accurate position description to recruit a person to fill this position.

Biddle and Thomas edited and published the first book on role theory in 1966. They defined roles as sets of prescriptions defining what the behavior of a position member should be [13]. A role should be a collection of rights and duties. They conclude that role theory is realistic and possesses an identifiable domain of study, perspective, and language [13].

In 1980, Bostrom performed a field study for role conflict and ambiguity from the viewpoint of users and designers of Management Information Systems (MIS) [14]. A role is defined as a set of expectations about behavior for a particular position within a system. Generally, a role is a position occupied by a person in a social relationship. At this position, the person possesses special rights and takes special responsibilities. It is concluded that an inadequate role-defining process is a key source
of role problems for users and designers to obtain satisfaction of working with MIS.

Zigurs and Kozar conduct a field study on the requirements and usage of roles from the viewpoint of users of Group Support Systems (GSS) [99]. They review the literature on roles from 1930s to 1990s including [12] and discuss the user behavior and influence to the role expectations of GSS users. The GSS used in their study has “only software that collects, manipulates and aggregates users’ inputs” and does not support role mechanisms. The roles of users are completely up to the organization of the user groups. They define a role as “a dynamic set of recurring behaviors, both expected and enacted, within a particular group context”. King and Sethi study the impact of socialization on role adjustment of information system professionals [48]. Their work is motivated by the socialization process relevant to roles such as role orientation, role ambiguity and role conflict mentioned in the literature of sociology and psychology. It is concluded that institutionalized socialization tactics lead to a custodial role orientation, individual socialization tactics produce an innovative role orientation, and institutionalized tactics reduce role ambiguity and conflict among new employees. This conclusion demonstrates a clear requirement for an information system to be equipped with clear role mechanisms.

Ashforth did a comprehensive study and published his book on role transitions in 2001 [7]. It is a good addition to the role theory literature in social psychology. He cites the role definition from [13]. According to [7], a role is defined as a position in a social structure. A position means a more or less institutionalized or commonly expected and understood designation in a given social structure such as accountant (company), mother (family), and church member (religious organization). All his discussions on roles are based on identities of people.

Acuna and Juristo apply role concepts intuitively in software project management [1]. They concentrate on human resource management in a software development and believe that the theory of psychology helps software project management. They define roles as sets of responsibilities and capabilities required to carry out the activities of each subprocess. A capability defines the skills or attributes of a person. This method is capability-oriented and roles are taken as elements of a software engineering process.

For example, a development group and a data management group are faced with complaints that the results presented to customers are not based on the correct source data customers assume to be. Each group thinks that the other group is responsible for quality assurance. In building their role model they can easily agree on the need for a quality assurance role. As they flesh out the responsibilities of that role it becomes clear that it has to be filled by a team with representatives from both organizations. The team is then further described in terms of the roles (and responsibilities) of its members.

In business management, role modeling (RM) is proposed as a business engineering technique [44] to provide a model of an organization in terms of roles, responsibilities and collaboration among individuals and teams. It also provides a process applicable in both a small and large scale for an enterprise to transform and advance and a vehicle for reengineering and process improvement. In RM, a role has a cohesive set of responsibilities and a purpose for its existence. A role may be played by a person, group, organization, team, or automated system. Each of these entities may fill many roles. In fact, a role may be filled by another role. RM draws a major part of its power by separating “what” of a role from “who” is filling the role at any particular point in time. That separation allows teams to come together to discover the nature of the work that must be done to meet the demands placed upon them by their sponsors, and then to figure out how the roles must be filled.

As discussed above, in social psychology and management, the major research is on how people behave under the meaning of roles played by people. The relevant problems about roles in human’s behavior include role description or specification, expectation, conflicts, and transitions. All these problems largely affect the performances of a group of people [7, 13, 14, 48, 99]. In reality, it is difficult to describe roles clearly and strictly because natural languages are ambiguous to some extents. That is why different persons at the same position can make different contributions and human resource experts are required to describe a position as clearly as possible. All their findings assure the importance in supporting roles in information systems (Table 6).

### Table 6 Social-Roles

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Motivations</th>
<th>Contributions</th>
<th>Conclusions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1966</td>
<td>Biddle Thomas</td>
<td>To propose and establish the role theory</td>
<td>Publish the first book that systematically discusses different aspects of role theory</td>
<td>Roles can be taken as a concept to describe and facilitate a person’s behavior and jobs.</td>
</tr>
<tr>
<td>1980</td>
<td>Bostrom</td>
<td>To clarify the relationships between users and MIS designers</td>
<td>Discuss role conflict and ambiguity in MIS design</td>
<td>An inadequate role-defining process is a key source of role problems in MIS design.</td>
</tr>
<tr>
<td>1994</td>
<td>Zigurs Kozar</td>
<td>To explore and describe the behavior of users and designers of GSS</td>
<td>Perform a field study on the requirements and usages of roles from the GSS users</td>
<td>Task and group-building roles are required to achieve the GSS effectiveness and participants’ satisfaction.</td>
</tr>
<tr>
<td>1998</td>
<td>King</td>
<td>To clarify the impact of socialization on role adjustment</td>
<td>Survey MIS professionals on the relationships between organizational socialization tactics and adjustment variables like role conflict and clarity</td>
<td>The socialization practices affect a professional’s role adjustment. Employees’ institutionalized tactics reduce role ambiguity and conflict.</td>
</tr>
<tr>
<td>2001</td>
<td>Ashforth</td>
<td>To describe organizational life with a special perspective</td>
<td>Publish a book that discusses comprehensively role transitions in organizational life</td>
<td>Role transitions are normal in an organization and each role has an identity including goals, values, beliefs, norms, interaction styles and time horizons.</td>
</tr>
<tr>
<td>2004</td>
<td>Acuna</td>
<td>To improve human</td>
<td>Propose a procedure to assign team</td>
<td>The role assignment procedure can make it easier</td>
</tr>
</tbody>
</table>
8. CONVERGENCE OF ROLE CONCEPTS

To evaluate and compare the role concepts and mechanisms in different research and engineering areas in information systems, we need to consider and learn the role concepts from management and social psychology. We conceive that the ideas of roles in management and social psychology cover what a role should be, because an information system is a virtual community that simulates, certainly, sometimes surpasses a real one. Persons have two kinds of existence, i.e., server and client. When they play a role, they provide specific services and have specific rights to ask for services. With this common sense, a role can be considered as a view of persons on the world. When they play a specific role, they have a special view on the surroundings. Their role in a working environment is actually a wrapper with a service interface [35, 80] and request interface [67, 94, 96].

From what we have discussed, we find that much work develops and applies the role concepts in an ad hoc manner. We also find that there is no much relevance among many papers that should be related. To have a clear clue for the research and practice on roles, we create Table 7 to compare the meanings and usages based on the ideas in the relevant work on roles. We can conclude that Agent-roles and System-roles as an expansion of RBAC-roles are currently the most advanced ones.

<table>
<thead>
<tr>
<th>Table 7 Comparison among the Different Kinds of Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aspects</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>Reasons of why roles are applied</td>
</tr>
<tr>
<td>Role players</td>
</tr>
<tr>
<td>Role content</td>
</tr>
<tr>
<td>Role specification</td>
</tr>
<tr>
<td>Role assignment</td>
</tr>
<tr>
<td>Role relationship</td>
</tr>
</tbody>
</table>

4) System-roles are an expansion to accommodate more properties than RBAC-roles that aim to simplify the work of system administrators.

5) CSCW-roles aim to support people to collaborate with the help of computers.

This figure actually shows a classification with a class hierarchy of simple inheritance and whole inheritance from the viewpoint of object-orientation. CSCW, RBAC and modeling-roles should inherit the social-roles.

We can also understand different role concepts from various viewpoints in an information system. They can be categorized into four unique views of roles according to the information system layers as shown in Fig. 2. From the viewpoint of operating systems, we agree that the idea of roles is similar to that from RBAC. From the viewpoint of modeling and programming methodologies, we have modeling-roles. From the view point of applications and human users, we have CSCW-roles, agent-roles, and social-roles.
Collaboration, we need to pay the same attention to both services and requests \([94, 96]\). Roughly speaking, either modeling-roles or RBAC-roles take about a half of the social-roles. They together constitute social-roles. Agent-roles have considered \textit{rights} as part of a role, and can thus be improved by accommodating more generalized \textit{rights}, i.e., \textit{requests}. As a simulation of intelligent systems, roles in agent systems, or agent-roles, are approaching the ideas of social-roles.

9. ROLE-BASED SYSTEMS AND THEIR CHALLENGES AND BENEFITS

To clarify the challenges of role-based systems, we need to first clarify what we mean by role-based systems. By role-based systems, we mean that they are analyzed with role-based models, designed by role structures, and constructed with roles as first-order components. To build role-based systems, roles should be used as underlying mechanisms defined, specified, constructed and applied. In this sense, role-based systems have their own challenges and benefits which come from the current situations of the research of role mechanisms (Table 7).

In the sense of social-roles, roles can be used to facilitate collaboration among people using CSCW systems. To implement social-roles with computers can well support CSCW system designs. In role-based CSCW systems, roles can prevent people from being overwhelmed by too much irrelevant information. In the sense of modeling, agent and roles can be used to facilitate collaborations and interactions among systems, computers, subsystems, agents, and system components.

To implement real role-based systems, the problems to be solved include what roles are, how to express, present, store, change and reuse a role. To completely support role-based systems, we face many challenges.

9.1 Challenges

From the viewpoint of social psychology, although there is huge literature on role theory, there is no conclusion that how roles affect the satisfaction, effectiveness and results of collaboration. We need to conduct more studies to answer it. We may encounter such questions as:

- Do people enjoy working in an environment with roles clearly tagged?
- Do people enjoy working with process-roles or interface-roles?
- Do people enjoy working with people or with roles (people are hidden by roles)?
- Under what situations do people enjoy role-based collaboration?

From the viewpoint of CSCW system design perspectives, CSCW-roles cannot support well enough role content, role specification, role assignment, and role relationships. Innovative presentation tools are also required for collaboration facilitators. We need to consider the social and psychological requirements of human users and design roles that facilitate collaboration among people. We may encounter the following questions:

- How do we present roles with special design?
- How do we remind people of their roles?
- How do we evaluate the effectiveness of role-based collaboration?
• How do we support role negotiations including role assignment and role specification [96]?
• How do we build role hierarchy?
• How do we schedule people with different roles?

From the viewpoint of agent systems, agent-roles have not been well-defined to support role relationships and role content. We need to consider the ways to organize agents to form intelligent agent systems in order to make agents possess the properties such as being autonomous, autonomic, reactive, active, pro-active, and social and adaptable. We need to answer such questions:

• How can we build a real role grid for agents to live and grow (sociality)?
• How can we make roles as the dynamics of agents (autonomy)?
• How can we design agents to learn to play new roles (learning capacity)?
• How do we use roles encourage agents to actively work in an agent community (activity)?
• How can we make roles as easier environments for agents to adapt (adaptability)?

From the viewpoint of object modeling, current modeling-roles have not yet been well-designed to support complete role content, role assignment and role relationship. We need to develop roles into a mechanism similar to classes, objects, and functions. To make roles a first-class modeling mechanism, we need to answer the following questions:

• What should a role express in role specification?
• How should we make roles a well-defined mechanism that accommodates all the aspects related to roles?
• As for modeling languages, what language mechanisms should we provide to help specify roles?
• How do we extract roles from a problem domain, i.e., what is the methodology [95, 97]?
• How do we express and process role assignment in software development [95, 97, 98]?

9.2 Benefits

Benefits come with challenges. With computer-supported tools, roles can help their players (people, objects, agents, computers, systems, subsystems, system components)

• Identify the role player “self” [7];
• Avoid irrelevant interruptions that harm working performance [24];
• Enforce independency by hiding people under roles;
• Encourage people to contribute more;
• Remove ambiguities to resolve expectation conflicts [14, 41, 42];
• Work with personalized user interfaces;
• Concentrate on a job and decrease possibilities of conflicts for shared resources;
• Distribute tasks based on the overall requirement of a group;
• Decrease the workload of system administrators;
• Implement separation of concerns;
• Decrease the search space of knowledge;
• Create dynamics for agents; and
• Regulate easily the ways of collaboration among agents.

10. CONCLUSION

Roles have been widely applied for many years. There are many more applications to come. For example, in ontological approaches [5], roles are fundamental concepts to organize structures and relationships [39]; in agent systems [17, 20, 53, 60], agents’ behavior can be adjusted and tuned by playing different roles or by tuning their current roles; in systems of systems roles can be used to separate concerns in designing [43]; in exploiting the information web [37], roles are good mechanisms to extract relevant information; in human–robot systems [78], roles can be used to regulate the interactions between humans and robots; and in HCI, personalized user interfaces present information in ways compatible to roles [38, 79]. In CSCW, people can talk to others at a definite time if a role playing schedule is set. By this schedule, interventions are blocked or arranged to a special time slot. In management information systems, such as role-based workflow systems, one may do what the roles specify clearly by avoiding too much irrelevant information to improve the productivity. This paper concludes that:

1) Roles are important in information systems design but are often introduced intuitively and then applied.

2) Introducing roles in different research and practice fields has different motivations, aims, concentrations and goals. Roles can be taken as modeling mechanisms, management strategies, interaction media and behavior describers;

3) Obtaining the underlying commonalities of roles by referring to the ideas from management and social lives is required to promote the applications of roles and is an active trend in different research fields.

4) The research and practice on roles still face many challenges. We can foresee more interesting and successful activities in the research and practices of roles.

In applying the role concept, a key problem is how to specify a role and how to apply a role in a system. The traditional role mechanisms and role concepts are not qualified to accomplish these tasks. We need to develop new role mechanisms to support easy and clearer role specification. There are many essential topics still unexplored, requiring additional, comprehensive research.

1) Along the directions of both RBAC-roles and modeling-roles, we should find more applications, e.g., Computer-Aided Software Engineering (CASE) tools, personalized interfaces, business management, and Flexible Manufacture Systems (FMS) [94 - 97].

2) It is required to develop a well-defined role model to accommodate generalized requirements in different aspects, such as, role transfer, the dynamic structures of roles, and the roles for groups.

3) It is needed to investigate the mechanism of group roles, i.e., how to have a group play a role.

4) It is required to build a prototype system to fully support the social roles including agent-roles and CSCW-roles. CSCW-roles are required to concentrate on the interface aspect and agent-roles on the process aspect.
5) A role can be considered as a set of incoming and outgoing messages. We still need to find a practical and efficient way to form user interfaces based on this set of messages. More research is needed to provide a role specification tool. We need a more structural and efficient way to bind messages to a role.

11. ACKNOWLEDGMENTS
This research is supported in part by NSERC (National Science and Engineering Research Council, Canada, No. 262075-06) and the IBM Eclipse Innovation Grant Funding. We greatly appreciate all the reviewers’ constructive comments that motivate us to think more and improve the presentation of this paper.

12. REFERENCES
[8] Bachman, C.W. and Daya, M., "The Role Concept in Data Models", in Proc. of International Conference on Very Large Databases, Tokyo, Japan, 6-8 October 1997, pp. 464-476.


