

Improving Object-Oriented Analysis with Roles

Haibin Zhu, *Senior Member, IEEE*

Department of Computer Science and Mathematics, Nipissing University
100 College Drive, North Bay, Ontario, P1B 8L7, Canada

Email: haibinz@nipissingu.ca

Abstract – *Object-Oriented Analysis (OOA) has been proposed and applied in software engineering for more than fifteen years. Many researchers and practitioners have published many articles and books to discuss this methodology. Many companies and organizations have also published many standards in software development. However, there is one critical problem that is becoming the bottle neck affecting the efficiency and correctness of OOA, i.e., the gap between customers, analysts and designers is too large to obtain a satisfactory requirement analysis report. Role-based analysis is one possible way to alleviate this symptom. It can at least improve the quality of the requirement analysis report.*

This paper gives a retrospect to the past OOA methodology, analyzes the weakness of it, proposes how to practice a new method for system analysis, i.e., role-based analysis, and demonstrates the benefits of this method.

Keywords: Role, System Analysis, OOA, Class

1 Introduction

Cognitive Informatics (CI) is a multidisciplinary research area that tackles the fundamental problems shared by modern informatics, computation, software engineering, AI, cybernetics, cognitive science, and other sciences. The development and the cross fertilization between the aforementioned science and engineering disciplines have led to a whole range of extremely interesting new research areas relevant to CI. Software development is no doubt an important research branch that will improve the research of CI and also requires the support from the CI methodologies [2, 12, 20, 31, 32].

Software development is a difficult job in Information Technology (IT) industry. Even though many IT companies have declared for many years that they overpaid their software developers, the salaries of software developers obtained a 4% increase in 2006 [4]. In 2003, the author interviewed a mature student who was a founder and CEO of a small company. He told the author that he did not want to pay the high salary to hire software developers who could not do as what he wanted them to do. He thought that all the company's

income is mostly up to his imagination and his creative ideas. He also showed the author his blueprint of an internet application that seemed interesting. He believed that his new designs would bring profits for his company. Therefore, he wanted to enroll the Computer Science Program of Nipissing University to learn software development by himself. He also wanted to implement his design and ideas by himself. However, he failed to implement his promise to complete this program. After showing up several times, he did not come to the university again. There might be many reasons including financial, managerial and personal reasons. At least, one reason is that he cannot learn by himself. It is too hard for him to learn software development.

The above story demonstrates two aspects: one is that system analysis is so difficult that software developers' work is unsatisfactory to the customers and the project initiators; and the other one is that the software development is highly intelligent and still difficult for customers to master and DIY (do it yourself). Specially trained software developers are still required in software industry.

Analysis is "the resolution or breaking up of anything complex into its various simple elements, the opposite process to synthesis; the exact determination of the elements or components of anything complex (with or without their physical separation)" [19]. Analysis requires a substantial amount of effort. It should be undertaken after management has decided that the systems development project is under consideration. In this paper, analysis means the analysis phase of Development Life Cycle (SDLC) [6].

The most difficult part in software engineering is system analysis [22], because there are always gaps between customers (or requirement describers) and system analysts. It is very difficult to file a satisfactory requirement report for all the parties involved in requirement analysis. 10 reasons for information system projects' failure are almost all relevant to system analysis [23]:

- 1) Project managers do not understand users' needs.
- 2) The project's scope is not well-defined.
- 3) Project changes are managed poorly.
- 4) The chosen technology changes.
- 5) Business needs always change.

- 6) Schedules are not realistic.
- 7) Users do not welcome the project.
- 8) Sponsorship is lost and there is not enough financial support.
- 9) The project lacks appropriate skillful people.
- 10) Past practices and lessons are always ignored.

In system analysis, there are tasks, such as, information gathering, system requirements definition, requirements prioritization, feasibility and discovery prototyping, and alternatives evaluations [26]. All these activities are difficult and they need special approaches to obtain satisfactory results.

Object-oriented analysis (OOA) was proposed as a good method more than fifteen years ago and there are many publications in such a topic [11, 16, 20, 35, 38]. However, no convincing result has been obtained hitherto. Roles have been discussed for many years and many advantages have been acclaimed [5, 8, 9, 13, 28]. However, there is evident lack of investigation of system analysis with roles. This paper proposes role-based analysis (RBA) to improve OOA with roles and clarify why RBA can improve OOA.

This paper is arranged as follows: Section 2 reviews the key points of object-oriented analysis; Section 3 discusses the difficulty or the weakness of OOA; Section 4 propose an analysis process with roles, i.e., RBA; Section 5 compares OOA with RBA from the tasks of system analysis; Section 6 suggests some aspects to build tools to support RBA; Section 7 presents some related research; and Section 8 concludes the paper and proposes future research topics.

2 Object-Oriented Analysis

System analysis is a cognitive activity that to abstract from informal system descriptions to create formal ones. System analysis can occur at many different levels of abstraction. At the business or enterprise level, the techniques associated with OOA can be coupled with a business process engineering approach in an effort to define classes, objects, relationships, and behaviors that model the entire business.

At the business area level, an object model that describes the processing of a particular business area can be defined. At an application level, an object model focuses on specific customer requirements as those requirements affect an application to be built.

OOA emphasizes more on modeling the problem domain and acquiring the objects that exist in the problem domain. The objects can be tangible or abstract. OOA concerns the activities in the early stages of software development. In the early stages, interfaces, control strategies, infrastructure functions, human user types, and application scenarios are emphasized.

OOA tries to identify the type of objects that map into components of the problem to be solved. This activity is to find the major relationships between the different types of objects considered as class instances. Classes are defined with the data they manage, the services they provide, the constraints they comply with and how they relate to other classes. Classes must meet some functional requirements and usually reflect a certain system viewpoint.

The process of OOA is as follows [11, 16, 20, 35, 38]:

- 1) Identify classes: From the problem description, relevant entities and their relations should be found and then classes are formed.
- 2) Specify internal structures: Each class should provide a template for its instances with a well-defined structure. In this activity, specified classes are used.
- 3) Specify services: Every class should support a group of meaningful operations or services. Operations such as constructors, destructors, selectors and modifiers should be classified. The properties of minimalism, completeness, and convenience with regards to the services should be considered.
- 4) Specify dependencies: Classes are not independent and they have many relationships such as use, inheritance and composition.
- 5) Specify interfaces: Classes are taken as components of a system. They are the outputs of one group and inputs of another group. Interfaces are the service demonstrations provided by one group and applied by another group. Services should belong to different categories based on the different of group users. Functions should be divided into public operations or private services.

A good analysis method should have short cognitive distances [17] in its process. Cognitive distance tells the difficulty of a learning method for people to understand knowledge. The shorter the cognitive distance is, the easier the relevant method is. OOA is better than traditional structural analysis in that it has a shorter cognitive distance, because it is easier to find classes than to find procedures in a problem domain. Object-oriented analysis (OOA) is applying object-oriented methodology into the analysis phase of a Software SDLC [22]. In this phase, the object models of a real-world application are developed. The object models should specify the functional behavior of the system independent of implementation details. OOA is one tool and methodology to achieve this goal.

OOA's success is to view classes/objects as representation constructs [1, 3]. From the representation point of view, OOA supports abstraction, classification, existence, composition, and iteration.

3 The Weaknesses of OOA

The weaknesses of OOA are mainly located at the analysts' understanding of problem descriptions. Identifying a class is the most important activity in OOA. It is an abstraction process from many concrete objects to a generalized concept. It is also an induction or abstraction process from particular cases to a general truth, or from concrete to abstract. To accomplish this task, analysts must possess a very strong capability to abstract. This process is now largely based on manual work, or natural intelligence. In this step, analysts should make full use of their basic knowledge of problem solving learned from their previous educations and working experiences.

In OOA, analysts need to understand the problem clearly. This understanding requires analysts understand the domain knowledge that is time-consuming to fully master. That means analysts should pay more attention to the application. Lack of domain knowledge may lead to wrong abstractions. That is why different analysts may create different analysis reports for the same problem. Therefore, many analysts try to propose some methodologies that can be applied without knowing too much domain knowledge.

There are some proposals to state that in a problem description, usually nouns correspond to classes and verbs represent functions [10]. The problem of this method is that too many nouns and verbs are contained in a description document. Therefore, taking nouns as candidate classes is not practical. There are too many variations to which nouns or noun combinations are qualified to be taken as classes.

Suppose there are n nouns. The number of possible classes is actually the sum of the combination numbers selecting i from n ($i = 1, 2, \dots, n$), i.e., $\sum_{i=1}^n C(n, i) = 2^n - 1$. It means that the complexity of identifying classes is an NP-hard problem.

Some advices from the creator of C++ [30] might be helpful for us to find a class:

- "If you can think of "it" as a separate idea, make it a class.
- If you can think of "it" as a separate entity, make it an object of a class.
- If two classes have a common interface, make that interface an abstract class.
- If the implementations of two classes have something significant in common, make the commonalities a superclass."

However, in these guidelines, how could analysts express "separate idea", "separate entity", "common" and "significant" with programmable entities? There are still large cognitive distances for analysts to deal with. Therefore, it is still a highly intelligent task to

identify classes from ambiguous requirement descriptions, because there are too many possibilities of identifying classes from a written document or oral descriptions.

4 Role-Based Analysis

Roles are used to capture important properties of objects and provide useful information for how to act with these properties in a system [5]. Roles can be used to formalize the concept of a type that depends on the referencing relationships. In a system, an object may play several roles at a time. Roles are a concept or specialization of a concept. Roles have the functions in system analysis as follows [8, 9, 13, 28]:

- Model a perspective of a phenomenon;
- Specify a position and a set of responsibilities which are made up of services and tasks;
- Improve the precision of procedure interface specifications;
- Express precise referencing and interaction behaviors between objects;
- Express constraints on the coordinated movements of objects between data structures;
- Express the organizational structure of a multi-agent system; and
- Specify interactions in a generic way.

By role-based analysis (RBA), it is emphasized that roles are applied in system analysis as foundation mechanisms. Roles, compared with classes, have shorter cognitive distances in analysis for people to understand a system, because roles are natural architectural components of systems. To build a system, roles should be specified before class specifications [36]. Based on E-CARGO [34, 36, 37], a role is defined as a combined interface including incoming messages as a service interface and outgoing messages as a request one. This definition is appropriate for analysis, because the task of analysis is mainly finding the framework of a system. Therefore, specifying roles is an important task in system analysis.

4.1 The E-CARGO Model

To facilitate analysis, the E-CARGO model is slightly revised. A software system Σ can be described as an 8-tuple $\Sigma ::= \langle C, O, \mathcal{A}, \mathcal{M}, \mathcal{R}, \mathcal{E}, \mathcal{G}, s_0 \rangle$, where C is a set of classes, O is a set of objects, \mathcal{A} is a set of agents, \mathcal{M} is a set of messages, \mathcal{R} is a set of roles, \mathcal{E} is a set of environments, \mathcal{G} is a set of groups, and s_0 is the initial state of a system including the primitive components. The human set is removed from the original model to mean a pure software system.

An *object* is everything in a system that occupies a memory cell or cells. It can be accessed and its data or status can be changed by operations on it. To support analysis, it is emphasized that objects cannot play roles.

A *class* is a template for a group of similar objects. It describes the operations on the group of objects and specifies the data structure of these objects.

An *agent* is a special object that can play roles, i.e., role players. It is defined as $a ::= \langle n, c_a, s, \mathcal{N}_r, \mathcal{N}_g \rangle$, where, c_a is a special class that describes the common properties of users, n is the identification or name of the agent, s is the set of properties of the agent, \mathcal{N}_r means a set of identifications of roles the agent is playing, and \mathcal{N}_g means a set of identifications of groups the agent belongs to. C_a denotes the set of all agent classes.

A *message* is defined as $m ::= \langle n, v, l, \mathcal{P}, t \rangle$ where n is the identification of the message, v is null or the receiver of the message expressed by an identification of an object or an agent, l is the pattern of a message, specifying the types, sequence and number of parameters, \mathcal{P} is a set of objects taken as parameters with the message pattern l , and t is a tag that expresses any, some or all message.

A *role* shows both the rights/requests and responsibilities/services of human users. Incoming messages are used to express responsibilities and outgoing messages to express rights. It is defined as $r ::= \langle n, I, \mathcal{N}_a, \mathcal{N}_o \rangle$ where, n is the identification of the role, $I ::= \langle \mathcal{M}_{in}, \mathcal{M}_{out} \rangle$ denotes a set of messages, wherein, \mathcal{M}_{in} expresses the incoming messages to the relevant agents, \mathcal{M}_{out} expresses a set of outgoing messages or message templates to roles, i.e., $\mathcal{M}_{in}, \mathcal{M}_{out} \subset \mathcal{M}$, \mathcal{N}_a is a set of identifications of agents that are playing this role; and \mathcal{N}_o is a set of identifications of objects including classes, environments, roles, and groups that can be accessed by the agents playing this role.

An *environment* expresses a structure to build a group. It is specified by roles, the objects accessed by the roles and the cardinalities of agents playing roles. $e ::= \langle n, \mathcal{B} \rangle$ where

- n is the identification of the environment; and
- \mathcal{B} is a set of tuples of role, number range and an object set, $\mathcal{B} = \{ \langle n_r, q, \mathcal{N}_o \rangle \}$. The number range q tells how many users may play this role in this environment and q is expressed by [lower, upper]. For example, q might be [1, 1], [2, 2], [1, 10], [3, 50], It states that how many agents may play the same role r in the group. The object set \mathcal{N}_o expresses the objects accessed by the agents that play the relevant role. The objects in \mathcal{N}_o might be owned by one agent or shared by many agents. Sharing or not is specified by the relevant roles.

For example, a classroom is a typical environment for teaching can be expressed as

- $n = \text{classroom}$;
- \mathcal{B} is as follows: one teacher and 5-40 students expressed by $\{ \langle \text{teacher}, [1, 1], \{ \text{the room, a}$

white board, a brush, a set of color erasable markers, a computer, a data projector and a pointer} \rangle, \langle \text{student}, [5, 30], \{ \text{the room, 30 desks and 30 chairs} \} \rangle.

A *group* is a set of agents that are working on an environment, i.e., a set of agents assigned with roles in the relevant environment.

4.2 Analysis with E-CARGO

E-CARGO gives a guideline to analyze a system with roles. The proposed RBA is in fact an analysis methodology with the E-CARGO model. In E-CARGO, agents are emphasized as role players. Objects are accessed through roles by agents. Objects do not play roles.

Specifying roles is actually to collect public message templates that can be implemented by agents, i.e., the service interface and the necessary one provided by other agents, i.e., the request interface. All operations in the service interface of a role should be accomplished by one agent classes. All the operations in the request interface of a role should be mapped to one or more other roles (Fig. 1) [34]. In Fig. 1, \square is used to express a role, \bigcirc an agent, \longrightarrow the requests of a role, $\bigcirc \longrightarrow$ the services of a role.

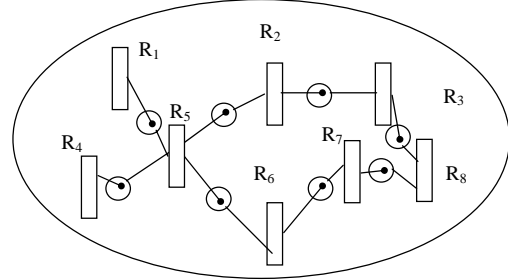


Fig. 1. The requests and services among roles.

Because the model E-CARGO includes classes and objects, the key advantages of OOA are kept in the RBA to allow analysts to change and tune specifications. As for the E-CARGO model, the major concerns are environments (\mathcal{E}), roles (\mathcal{R}), and agent classes (C_a) in RBA. The following activities may be iterative:

- 1) Identify and specify environments: From the problem description and domain knowledge, the environments are found and specified including roles, cardinalities and objects, i.e., $\langle n_r, q, \mathcal{N}_o \rangle$.
- 2) Specify roles: From the problem description and relevant domain knowledge, the requests and services of roles are specified.
 - a. Specify the outgoing messages (requests) including the message receivers, message patterns, parameters and tags, i.e., $\langle v, l, \mathcal{P}, t \rangle$; and

- b. Specify the incoming messages (services) including the message patterns, parameters, parameters and tags, i.e. $\langle l, p, t \rangle$.
- 3) With the help of the roles and environments, OOA tasks are performed:
- a. Identify and specify object classes: find and define classes that describe objects to be mainly accessed.
 - b. Identify and specify agent classes: find and define classes that describe agents that play roles.
 - c. The classes (including object classes and agent classes) may be arranged into inheritance hierarchies and composition relationships.

In RBA, the major help to identify classes is that classes should cover roles. That is to say, $\forall r \exists c \ni c.X \supseteq r.M_i$, i.e., for every role, there must be an agent class that is qualified to play it. Roles are not classes. That means candidate agent classes must cover the specified roles.

In the above methodology, there are two kinds of classes to be specified. One kind of class is mainly used to express the passive objects that do not play roles, called object classes; the other kind is mainly used to express active objects that can play roles, called agent classes. These two kinds of classes have different functions in system analysis.

This is beneficial to release the argument between agents and objects in object-oriented and agent-oriented analysis. In the E-CARGO model, it is clear that objects and agents are different. Objects cannot be used to play roles. Agents are special objects that can play roles. In the implementation with object-oriented languages, roles may have no concrete entities to reside in the software system and the final product may have no roles in it. This means that roles are abstract analysis tools in analysis and design.

When an environment is specified, it may be needed to introduce new roles; when a message of a role is specified, it may be needed to introduce new classes. This requires analysts to reconsider the services and requests of roles and re-define the relevant environments and roles.

5 Comparison between finding classes and finding roles

RBA can be compared with OOA according to the tasks of system analysis [26]. In performing each task, the benefits or drawbacks of RBA and OOA can be clarified.

In system analysis, the first task is to *gather information*. The analysts should know how to complete this task. However, OOA does nothing in

completing this task. In *information gathering*, analysts should know whom they should contact. Here, it is easy to know “whom” from roles but it is difficult to know “whom” from classes. For example, analysts know that a manager has a more general view to the scenario of the system; a technical staff concerns more about the advances of technologies; and a user will use the system daily. They can collect different information from these different roles. There might be such argument that use cases outline the scenarios relevant to people. In fact, this demonstrates that OOA lacks facilities to accommodate aspects relevant to people. Use cases are methods that alleviate the symptoms of original OOA’s lacking expressions of aspects relevant people. Compared with RBA, use cases are not consistent with the specification of classes. However, roles are the basis for specifying classes. The specification of roles largely helps specifying classes.

In *defining system requirement*, RBA can be compared with OOA by the easiness and efficiency to obtain classes and roles.

[Example 1]: The following is a description for a simple problem of purchase management [10]:

“When ordering new videotapes from a supplier, the store manager creates a purchase order, fills in the date, the supplier’s name, address, and enters a list of videotapes to be ordered. The purchase order is added to a permanent list of purchases. When one or more video tapes are received from a supplier, a clerk locates the original purchase order and makes a record of each tape that was received. A record of the videotape is then added to the store’s inventory. When all tapes listed on a particular purchase order have been received, the manager sends a payment to the supplier and the purchase order is given a completion date.”

In such a short description of 111 words, there are 12 nouns, such as, videotape, supplier, order, clerk, recorder, store, date, name, address, list, inventory, and payment. The possible number of classes is $2^{12}-1 = 4095$.

Table 1 Roles in a store

| Role Names | Requests /Rights | Services /Responsibilities |
|-------------------|---|-----------------------------------|
| <i>Buyer</i> | Choose from the video tapes in store | Pay for the purchase |
| <i>Clerk</i> | Access the video tapes in store | Access the system interfaces |
| <i>Client</i> | Request video tapes | Pay to the store clerk |
| <i>Manager</i> | Access the video tapes in the store | Pay the clerks |
| <i>Supplier</i> | Access the video tapes to be provided, Request payment from the store owner | Provide video tapes to the store |

But if we extract roles from this description, we find that only three words are qualified as roles: *supplier*, *clerk* and definitely *manager* that is not shown in the description.

The possible number of classes is $2^2-1=3$. This demonstrates that there should be three agent classes to be identified. This significantly improves the efficiency to find or identify classes. Based on the specified agent classes, there are clear clues to identify and specify object classes.

The environment can be specified as: $e = \{<supplier, [1, 5], \{video\ tapes\}>, <clerk, [1, 3], \{the\ store\}>, <manager, [1], \{the\ store\}>, <buyer, [1, ?], \{the\ video\ tapes\ in\ the\ store\}>\}$, where “?” means that the upper bound of the cardinality can be very large.

After the environment is specified, besides agent classes, object classes are also specified, such as, *video tapes*, and *the store*. To specify the services and requests of these roles, domain knowledge is referred (Table 1).

[Example 2]: Another similar description from [27] is as follows: “Sally’s software shop buys software from various suppliers and sells it to the public. Sally stocks popular software packages and orders others as required. Sally extends credit to institutions, corporations, and some individuals. Sally’s software shop is doing well, with a monthly turnover of 300 packages at an average retail cost of \$250 each. Despite her business success, Sally has been advised to computerize.” This description is evidently inadequate for analyzing. However, this is a normal beginning for many clients to describe their requirements.

If OOA is used, classes are needed to be specified from nouns. The possible classes might be shop, software, public, package, credits, institution, corporation, individual, and turnover. The possible class number is $2^9-1=511$.

With E-CARGO, even though there are not evident words for roles except for “supplier”, an environment for a shop can be specified by the domain knowledge (Table 2).

$e = \{<Manger, [1], \{The\ shop\}>, <Casher, [1, 5], \{The\ shop\}>, <Accountant, [1, 2], \{The\ shop\}>, <Supplier, [1, 20], \{Software\ products\}>, <Buyers, [1, ?], \{Software\ products\}>\}$, where “?” means that the upper bound of the cardinality can be very large.

This is a regular software retail shop and typical domain knowledge will apply to it. Therefore, it is much easier for analysts to specify roles at first and then specify classes than to specify classes directly. Therefore, roles are superior to classes in *defining system requirement*.

Let see the third task, *prioritization of requirements*. This task again concerns more about roles. The analysts should contact users, technical staffs, and managers to balance the different priorities of

different functions. Generally, relationships among roles specify the priorities of requirement items. For example, a requirement emphasized by a manager should be assigned a higher priority in architecture. But a requirement emphasized by a user should be assigned a higher priority in interface design. Without roles, it is difficulty to prioritize the different requirement items.

Table 2 Roles in a software shop

| Role Names | Requests /Rights | Services /Responsibilities |
|-------------------|--|---|
| <i>Accountant</i> | Access the shop’s budget and cash flows. Ask the shop manager to approve the budget | Audit the shop; Bookkeep the inventory; Provide the investment advice; Payroll management; Process taxation |
| <i>Buyer</i> | Choose from the products in the shop; Request a software product | Pay for the purchase |
| <i>Clerk</i> | Access the software products in shop | Answer questions; Provide recommendations to purchases |
| <i>Manager</i> | Access the software products in shop | Responses to the requests from clerks |
| <i>Supplier</i> | Access the software products to be provided; Request payment from the shop owner | Provide video tapes to the store |

In the fourth task, *prototyping the feasibility and discovery*, OOA is a little better to use because object-oriented languages support classes directly and it is easier to obtain a prototype with OOA. RBA has now no available tools to compete with OOA at this step even though there are some trials to extend object-oriented languages to support roles.

The last task is *generating and evaluating alternatives*. This task requires the analysts to meet different roles to create requirement alternatives, because different roles consider different aspects for requirement alternatives.

The comparison can be shown in Table 3, where “+” is better “-”. From the comparison, RBA has high potential to thoroughly surpass OOA. Many benefits of RBA come from the domain knowledge implied in the role concepts.

Table 3 Comparison between RBA and OOA

| Methods \ Task | RBA | OOA |
|---|-----|-----|
| Information gathering | + | - |
| Defining system requirement | + | - |
| Prioritization of requirements | + | - |
| Prototyping the feasibility and discovery | - | + |
| Generating and evaluating alternatives | + | - |

6 Tools for Analysis with Roles

To better system analysis with roles, some tools are required. At first, a data base for roles, or called role base, should be built in which the services and requests of roles and their explanations are stored. Second, environments for different domains should be defined into a knowledge base or a data base. These tools are feasible because the role space is largely smaller than the noun space in a language.

To build role bases, it is needed to

- 1) List all the words and their synonyms relevant to the domain.
- 2) Specify the services and requests required for each role.

To do RBA with role bases, it is needed to

- 1) Search in the role bases those words from the requirement document, survey answer sheets, and oral talk notes. The words that can match the words in the relevant role base will be roles to be developed in the new system.
- 2) Clarify the services and requests of each role.
- 3) Design classes that can play the roles extracted in step 3, i.e., specify concrete functions or methods to implement the services specified in the role base.

A simple example of a role base is shown in Table 4 [14].

Environments are domain-oriented, to obtain an environment, the analysts can search the domain library for a typical environment to adjust. For example, the following environments are for different domains:

In a meeting, there are roles such as, Chairperson (Leader), Secretary (Recorder), Member, Presenter, Facilitator, Timekeeper, and Observer. That is to say, an environment for a meeting is {<Chairperson, [1,1], {the room, the tables, the chairs, the computer for chairperson}>, <Secretary,[1,1], {the room and the tables, the chairs, the computer for secretary}>, <Member, [3, 10], {the room and the chairs for

members}>, <Presenter, [1, 5], {the room, the computer and the projector for presenter}>, <Facilitator, [1,2], {the room, all the facilities in the room}>, <Timekeeper, [1,1], {the room, the timer}>, <observer, [1, 20], {the room, chairs}>}.

Table 4 Roles in system analysis

| Role Names | Requests /Rights | Services /Responsibilities |
|----------------------|--|---|
| Analyst | Interview stakeholders | Create formal analysis reports |
| Stakeholder | Request requirements | Participate in interviews and answer interview questions |
| Sponsor or supporter | Access the budget | Review problem statement |
| Architect | Review analysis reports | Attend peer review and suggest technologies |
| Project manager | Access the budget, the schedule, and the resources including human and equipment | Assist in developing the statement of work, provide updates to problem statement and manage resources |

In a software development team there are project managers, system analysts, system designers, programmers and testers. An environment is specified as {<Project Manager, [1], {All the facilities of the whole project}>, <System Analyst, [1, 3], {The analysis report}>, <System designers, [1, 5], {The analysis report, the design report}>, <Programmer, [1, 10], {The design report, the source code}>, <Tester, [1, 5], {The executable code}>}.

In an information system, there are many roles that are not played directly by people. The roles may played by agents. To find roles from a text or oral talk notes, it is needed to clarify what kinds of words can be taken as roles.

Conceptually, role is founded but not semantically rigid [9]. “Founded” means that a role can exist essentially independently. “Not semantically rigid” means that a noun cannot clearly specify the identity of its instances. Evidently, some nouns that exist depending on others should not be roles. For example, “a purchase order” is not a role, because it depends on a purchase. “Supplier” is a role because it has no direct instance and is independent. This criterion is too generalized to establish the real practical rules for role extractions from texts. It is needed to check the feature of words that can be used as role names. The following phenomena in English can be used to build the role base:

- Verbs are not for roles but for messages in roles;
- Adjectives and adverbs are not for roles;
- Nouns that have direct instances are not for roles, such as, dog, rabbit, pig, and cat.
- Words or phrases ending with “er” are good candidates for roles, such as, teacher, book keeper, engineer, and manager;
- Words or phrases ending with “or” are good candidates for roles, such as, operator, compressor, escalator and governor.
- Words or phrases ending with “ist” or “yst” are good candidates for roles, such as, analyst, socialist, and altruist.

7 Related Research

Although roles have not clearly acclaimed to be applied in system analysis, they are applied in many activities relevant to system analysis [4], especially in process specification.

In this category, roles are taken as concepts or tools to specify processes in mapping problem domain to solution one. They have different concerns on roles. Most of the contributions discussed here take roles as concrete processes.

Holt pioneered in the research on role activity theory [7]. The theory was improved by Ould who proposed role activity diagrams (RAD) to describe software processes [18]. 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 [15]. Various resources and events are associated with a role. A process involves the concurrent activity of several roles at one level. RADRunner [25] 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 [28]. 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; 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.

Reenskaug et al. apply roles to describe object-oriented software engineering processes [24]. 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 [31]. 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 implementations. Their work again demonstrates that a class-based programming language can be expanded to support roles with evolution requirement.

All the above contributions introduce roles in analyze system requirement by specifying specific process according different roles in a system. The limits of these contributions are located at taking roles as processes. In fact, roles as interfaces [29] are more relevant to system analysis.

8 Conclusions

Roles are a good mechanism to help improve system analysis in both efficiency and exactness. RBA is surpasses OOA in most aspects. However, there is a lack of comprehensive research in role-based analysis. To apply role mechanisms into system analysis requires a role base.

To claim the discussions of this paper and make RBA practical, the future topics need to be comprehensively investigated:

- Extract roles from different domains;
- Specify roles with both service interfaces and request interfaces;
- Build roles bases for different domains; and
- Build an analysis tool based the role bases to help requirement analysis.

Acknowledgement

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

References

- [1] Booch, G., Maksimchuk, R. A., Engle, M.K., Young, B.J., Conallen, J., and Houston, K.A., *Object-Oriented Analysis and Design with Applications (3rd Ed.)*, Addison-Wesley, 2007.
- [2] Chan, C., Kinsner, W., Wang, Y., and Miller, M. (eds.), *Proceedings of the 3rd IEEE International Conference on Cognitive Informatics (ICCI'04)*, Victoria, Canada, IEEE Computer Society Press, Aug., Los Alamitos, CA, 2004.
- [3] Coad, P., and Yourdon, E., *Object-Oriented Analysis (2nd Ed.)*, Prentice Hall, 1991.
- [4] Collett, "Salary Survey 2006: Hot Skills, Hot Pay", *Computer World*, November 13, 2006. Available: http://www.computerworld.com/action/article.do?command=viewArticleBasic&taxonomyName=careers&articleId=269951&taxonomyId=10&intsrc=kc_top.
- [5] Demsky, B. and Rinard, M. "Role-Based Exploration of Object-Oriented Programs", *Proceedings of the 24th International Conference on Software Engineering (ICSE'02)*, May 19-25, 2002, Orlando, Florida, USA., pp. 313- 324.
- [6] Hoffer, J.A., George, J.F., Valacich, J.S., *Modern Systems Analysis and Design (3rd Ed)*, Prentice Hall, New Jersey, 2002.
- [7] Holt, A. Ramsey, H.R. and Grimes, J. D., "Coordination System Technology as the Basis for a Programming Environment". *Electrical Communication*, vol. 57, no. 4, 1983, pp. 307-314.
- [8] Genilloud, G. and Wegmann, A., "A Foundation for the Concept of Role in Object Modelling", in *Proc. of the 4th International Enterprise Distributed Object Computing Conference (EDOC2000)*, Makuhari, Japan, Sept. 2000, pp. 76-85.
- [9] Guarino, N., "Concepts, Attributes and Arbitrary Relations: Some Linguistic and Ontological Criteria for Structuring Knowledge Bases", *Data & Knowledge Engineering*, vol. 8, 1992, pp. 249-261.
- [10] Irvine, K. P., *C++ and Object-Oriented Programming*, Prentice Hall, 1997.
- [11] Kaschek, R.; Mayr, H.C., "A Characterization of OOA Tools", *Proceedings of the Fourth International Symposium on Assessment of Software Tools*, 22-24 May 1996, pp. 59 – 67.
- [12] Kinsner, W., Zhang, D., Wang, Y., and J. Tsai, J. (eds.), *Proceedings of the 4th IEEE International Conference on Cognitive Informatics (ICCI'05)*, UCI, California, USA, IEEE Computer Society Press, Aug., Los Alamitos, CA, 2005.
- [13] Kristensen, B. B. and Østerbye, K., "Roles: Conceptual Abstraction Theory & Practical Language Issues", *Theory and Practice of Object Systems*, vol. 2, no. 3, 1996, pp. 143-160.
- [14] Kulak, D. and Guiney, E., *Use Cases: Requirements in Context*, Addison-Wesley, New York, 2000.
- [15] Murdoch, J. and McDermid, J. A., "Modeling engineering Design Process with Role Activity Diagrams", *Transactions of the Society for Design and Process Science (SDPS)*, June 2000, vol. 4, no. 2, pp.45-65.
- [16] Nerson, J.M., "Applying Object-Oriented Analysis and Design", *Communications of the ACM*, vol. 35, no. 9, Sept. 1992, pp. 63-74.
- [17] Nooteboom, B., "Learning by Interaction: Absorptive Capacity, Cognitive Distance and Governance", *Journal of Management and Governance*, vol. 4, 2000, pp. 69–92.
- [18] Ould, M. A., *Business Processes: Modeling and Analysis for Re-engineering and Improvement*, John Wiley & Sons, 1995.
- [19] Oxford, *Oxford English Dictionary*, <http://www.oed.com/>, 2006.
- [20] Parsons, J. and Wand, Y., "Using Objects for Systems Analysis", *Communication of ACM*, vol. 40, no. 12, Dec. 1997, pp. 104 - 110.
- [21] Patel, D., Patel, S., and Wang, Y. (eds.), *Proceedings of the 2nd IEEE International Conference on Cognitive Informatics (ICCI'03)*, London, UK, IEEE Computer Society Press, Aug., Los Alamitos, CA, 2003.
- [22] Pressman, R.S., *Software Engineering: A Practitioner's Approach (6th eds)*, McGraw-Hill, Toronto, 2005.
- [23] Reel, J.S., "Critical Success Factors In Software Projects", *IEEE Software*, May/June, 1999, pp. 18-23.
- [24] Reenskaug, T., Lehne, O. A. and Wold, P., *Working with Objects: the OOram Software Engineering Method*, Pentice Hall, 1995.
- [25] Role Modellers Ltd, "A Better Way to Support Collaboration", <http://www.rolemodellers.com>, 2004.
- [26] Satzinger, J., Jackson, R.B., and Burd, S.D., *System Analysis and Design in a Changing World*, Course Technology, Boston, 2002.
- [27] Schach, S.R., *Object-Oriented and Classical Software Engineering (7th eds)*, McGraw-Hill, Toronto, 2007.
- [28] Steimann, F., "A Radical Revision of UML's Role Concepts", *UML 2000*, pp. 194–209.
- [29] Steimann, F., "Role = Interface: A merge of concepts", *Journal of Object-Oriented Programming*, vol. 14, no. 4, 2001, pp. 23-32.

- [30] Stroustrup, B, *The C++ Programming Language*, Addison-Wesley, 2000.
- [31] VanHilst, M. and Notkin, D., “Using Role Components to Implement Collaboration-Based Designs”, in *Proc. of ACM 1996 Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'96)*, CA, USA, 1996, pp. 359-369.
- [32] Wang, Y., Johnston, R.H. and Smith, M.R. (eds), *Proceedings of the 1st IEEE International Conference on Cognitive Informatics (ICCI'02)*, Calgary, Canada, IEEE Computer Society Press, August, Los Alamitos, CA, 2002.
- [33] Yao, Y., Shi, Z., Wang, Y. and Kinsner, W. (eds.), *Proceedings of the 5th IEEE International Conference on Cognitive Informatics (ICCI'06)*, Vol. I & II, Beijing, China, IEEE Computer Society Press, July, Los Alamitos, CA, 2006.
- [34] Zhu, H., “Separating Design from Implementations: Role-Based Software Development”, *Proc. of the 5th IEEE International Conference on Cognitive Informatics (ICCI'06)*, Beijing, China, July 17-19, 2006, pp. 141-148.
- [35] Zhu, H. and Zhou, M.C., *Object-Oriented Programming with C++: A Project-Based Approach*, Tsinghua University Press, 2006.
- [36] Zhu, H. and Zhou, M.C., “Role-Based Collaboration and its Kernel Mechanisms”, *IEEE Trans. on Systems, Man and Cybernetics, Part C*, vol. 36, no. 4, July 2006, pp. 578-589.
- [37] Zhu, H. and Zhou, M.C., “Supporting Software Development with Roles”, *IEEE Trans. on Systems, Man and Cybernetics, Part A*, vol. 36, no. 6, Nov. 2006, pp. 1110-1123.
- [38] Zhu, H. and Zhou, M. C., 2003. “Methodology First and Language Second: A Way to Teach Object-Oriented Programming”, *Companion of the International Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'03)*, USA, Oct., 2003, pp. 140-147.