On Modeling System-centric Information for Role Engineering

Dongwan Shin and Gail-Joon Ahn
Dept. of Software and Information Systems
College of Information Technology
University of North Carolina at Charlotte
Charlotte, NC 28223, USA
{doshin, gahn}@uncc.edu

Sangrae Cho and Seunghun Jin
Dept. of Information Security System
Electronics and Telecommunications
Research Institute
Taejon, 305-350, South Korea
{sangrae, jinsh}@etri.re.kr

ABSTRACT

In this paper we present an approach to modeling system-centric information in order to facilitate role engineering (RE). In particular, we first discuss the general characteristics of the information required in RE. Afterwards, we discuss two informational flow types among authorities involved in RE process, *forward information flow (FIF)* and *backward information flow (BIF)*, together with the introduction of an information model which is greatly suitable for use in the backward information flow. System-centric information is incorporated in the information model and UML extension mechanisms are exploited for modeling the information. Not only can the information model provide those different authorities with a method for both analysis of resources and communication of knowledge in the RE process, but it can also help lay a foundation for successful implementations of RBAC.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*access controls*; K.6.5 [**Management of Computing and Information System**]: Security and Protection

General Terms

Security

Keywords

Role-based access control, Role engineering, Information Flow, Information model, Attributive permissions

1. INTRODUCTION

The concept of using roles in access control has attracted considerable attention in computer security communities over the last few years. Thus many formalized and practical approaches to recognizing the value of their usage in access control have been taken by both researchers and practitioners. The reference models have been proposed for role-based access control (RBAC) in [14, 15], and the efforts to extend the models, for example in the area of constraint specification, role administration, and role delegation, have followed in [1, 13, 20]. With those formal models as its quintessence, RBAC has grown to be a proven solution for managing access control in a simple, flexible, and convenient manner. RBAC has been also implemented successfully in a variety of systems or applications as user authorization services. For instance, it is used in a simple web-based application in order to provide users with different grades of services, or it is deployed in an entire operating system as an alternative to the *all-or-nothing* superuser model. In addition to those individual systems or applications, enterprise-wise large-scale systems also fall in the sphere of RBAC's influence. Enterprise users generally need to access multi-vendor and multi-layered applications and systems of which enterprise resources consist. Schemes like juxtaposing *enterprise-wise roles* with localized roles, which are effective only in the individual applications or systems, have been identified with the purpose of fertilizing RBAC's applicability into the enterprise-wise authorization boundary [8, 18]. In those schemes, simply put, enterprise users are authorized according to their membership of *enterprise-wise roles* so as to access multi-vendor and multi-layered enterprise-wise applications and systems.

However, there are still some important issues that need to be addressed more in RBAC, in particular from the perspective of its implementation. Role engineering (RE) is one of them. RE is an approach to defining roles and assigning permissions to roles [2], and thus enables to identify and build explicit objects used in access control from the implicit existence of roles within an organization. Since being a multi-disciplinary and cooperative process where different authorities usually are involved, RE calls for both a large amount of information to be exchanged and so many actions to be taken by those authorities with a view to sharing their knowledge and expertise for the process. Considering this, RE may be costly and time-consuming. For example, assuming that `Purchasing manager` role is to be engineered, HR department, an information security group, or a system administrator group may need to partake in the process by first investigating the semantics of `Purchasing manager` in their *own* domain, exchanging or discussing their documented semantics of `Purchasing manager`, and mapping those semantics each other. Moreover, it may be continuous and iterative process in order to reflect the modification and refinement of the semantics. Considering the number of roles to be defined in enterprise-wise environments (encompassing security policies such as separation of duties, IT resources, and manpower), it is certain that tremendous amounts of work are required of those authorities. In this context, RE poses challenges of a new dimension.

In order to address the problem of RE which has been depicted previously in a somewhat general manner, both researchers and practitioners have proposed several methodologies for RE. However, they often fail to discuss 1) RE from the perspective of systems to be protected and 2) how the information, which is used for the purpose of sharing expertise or eliciting artifacts regarding organizational security policies and systems in general and for defining roles with assigned permissions in specific, can be modeled. This situation can be partly attributed to the fact that it is essentially a requirement engineering process, and therefore it can be specified at different levels of abstraction and varying contexts. Nevertheless, it is important to have a good knowledge of systems since defined roles cannot function in isolation from the systems in which they are utilized. Hence, RE has to deal with a well-defined system-level view.

According to [12], which discusses the close relationship between RBAC roles and business processes, the analysis of business processes can work as a catalyst in the derivation of RBAC roles and permissions within an organization. The system-level view in the *solution domain*, which can be described both at a much higher level of abstraction and with less amount of information than that of objects and operations, can be very useful in RE to assist with the general understanding of RBAC roles and permissions in conjunction with business processes. The system-level information can also provide a good communication of technical knowledge for the authorities involved in RE which need to be often competent to deal with the technical issues. This holds true for especially those (an information security group in the previous example) whose primary expertise is to manage the high-level organizational security policies which are considered to be in the *problem domain*.

This paper focuses its attention into an approach to modeling system-centric information in order to facilitate RE process. In particular, it first discusses the general informational characteristics in RE and two informational flow types among authorities involved in RE, *forward information flow (FIF)* and *backward information flow (BIF)*. Then, it introduces an information model which is greatly suitable for use in the backward information flow. System-centric information is incorporated in the information model and UML extension mechanisms are exploited for modeling those information. UML is a widely accepted modeling language for software development, and we believe that it can also be a good candidate for visually modeling non-software entities such as information systems.

The rest of this paper is organized as follows. Section 2 discusses the various aspects of roles and RE within the context of a RBAC reference model (*RBAC96*) [14]. The prior research related to our work is also described in that section, together with the discussion of three different RE models. Section 3 discusses our approach to modeling system-centric information for RE. Then we discuss how to model the information in the UML language in Section 4. An example to demonstrate how the information model can be used in RE process is also described in the section. Section 5 discusses our experience and future research directions and concludes.

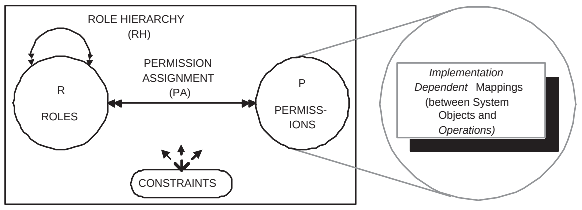

ROLE HIERARCHY
(RH)
PERMISSION
ASSIGNMENT
(PA)
R
ROLES
P
PERMISS-
IONS
CONSTRAINTS
Implementation Dependent Mappings (between System Objects and Operations)

Figure 1: Scope of RE in the context of RBAC96 model

2. ROLE ENGINEERING

Role engineering concerns capturing and analyzing various components of RBAC models for a later implementation. This requires a sound understanding of RBAC models and components. In this section we discuss RE within the scope of RBAC96 model. Then we discuss the prior works in RE and classify them into three RE models.

2.1 Scope within RBAC96 Model

The scope of RE in the context of RBAC96 model is described in Figure 1. In the RBAC reference model, activities in RE concern components such as roles, role hierarchies, permissions, permission assignment, and constraints.

While the meanings of roles are different, depending upon the context of their usage, they generally represent a set of competency and responsibility pairs [6]. In the reference models of RBAC [14, 15], they describe the relationship between users and permissions. Roles are used as a middle layer in between users and permissions. Users are human beings and permissions are a set of many-to-many relations between objects and operations. Roles bring users and permissions together, representing the job functions or titles and making it easy to apply organizational policies to the job functions or titles. Roles also decouple users and permissions, thereby reducing administrative routine works.

More advanced notions of roles include their hierarchical and constrained existence. The reference models of RBAC discuss those aspects of roles as well. Roles can be hierarchically structured so as to describe the lines of competencies and responsibilities within an organization. In the hierarchical structures, senior roles generally inherit the permissions assigned to junior roles, and this enables the role layer to be multi-layered, thereby further reducing the number of relations between users and permissions. Roles must be constrained in their relations to users and permissions as well as in the role-hierarchies. Constraints are an essential construct needed for laying out higher-level access control policies within an organization. A well-known example of constraints is the separation of duty. For instance, the same user cannot be a member of roles in a conflicting role set such as `purchasing manager` role and `accounts payable manager` role. The separation of duty constraint reduces possible frauds or errors by controlling membership in, activation of, and use of roles as well as permission assignment.

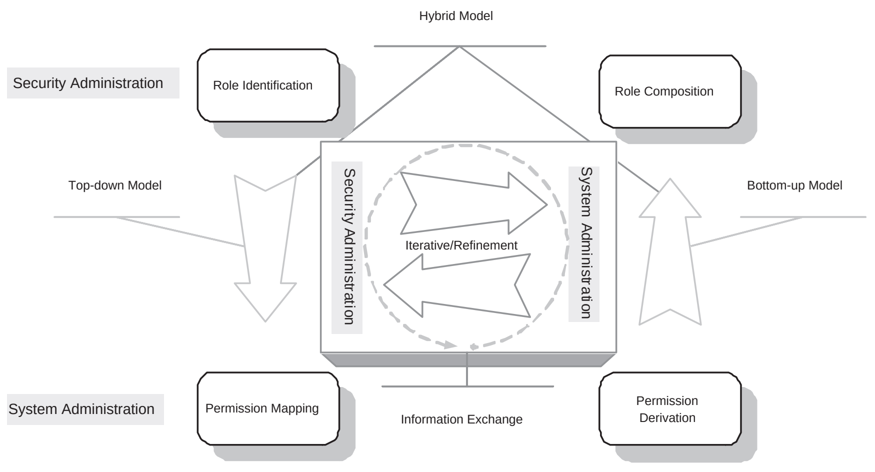


Figure 2: Models and information exchanges in RE

As stated in the previous section, RE is to define a valid set of roles and role hierarchies with associated permissions, though the level of abstraction of permissions may vary. By valid we mean that roles and role hierarchies must be constrained according to organizational security policies and also have properly designed semantics within an organization's RBAC environment. In order to fully leverage the concept of using roles in access control and benefit from the features of RBAC reference models, it is critical to consider RE to be of paramount importance and discretion. The invalid definition of roles resulting from RE can lead to both the defects in RBAC implementations and security breaches in its enforcement, and thus it may be quite hazardous and costly to detect and remove them at a later time.

2.2 Models and Related Works

There have been various approaches to engineering roles with the view of the implementation and administration of the roles[1]. They can be generally categorized into the following three models: *top-down*, *bottom-up*, and *hybrid* [4, 8]. Figure 2 shows the three models and describes the iterative nature of information exchange between two authoritative domains which are most likely to be involved in RE process, *security administration* and *system administration*. Security administration is in charge of organizational security (policy) management, whereas system administration concerns information system management *(see Section 3.1 for details)*.

2.2.1 Top-down Model

The top-down model can be generally described as an approach to deriving permissions from roles through the use of abstract concepts such as work-patterns and business processes [4, 12]. Looking at the model in some more detail, work-patterns or business processes in which roles are involved are analyzed and decomposed into smaller units in a functionally independent manner through role identification process. Afterwards, those smaller units or tasks are mapped onto permissions in information systems for their execution through permission mapping process. As shown in Figure 2, role identification falls within the scope of security administration, while permission mapping is in the range of system administration.

In [2] Coyne describes briefly how to identify roles in RBAC in a top-down manner. Taken as a whole, his approach uses system users' activities as a high-level of abstraction to identify candidate roles and remove duplicate candidate roles. Then permissions can be identified as a minimal set of access rights on the systems required to perform the roles. Finally, constraints definition follows before role hierarchies are built. Though introducing the concept of RE to the point, his approach lacks many technical details of RE process, only to be depicted in a highly conceptual manner.

In [12] Roeckle et al. discuss a process-oriented approach to finding roles in a top-down manner. The concept of *role-finding* is described for the purpose of deducting roles from business needs or functions. Their approach deals with an RBAC metamodel (we will discuss this in more detail in Section 4.1) to describe the notion of roles and their relation to users and access rights. Three different layers are discussed in conjunction with the metamodel: process layer, role layer, and access rights layer. Simply put, business processes are initially analyzed in the process layer. Then roles are derived from the business processes in the role layer and access rights on systems are assigned to the roles in the access rights layer. While their metamodel is well defined and structured, their procedural approach to finding roles lacks supporting some important issues such as how to handle role information update.

2.2.2 Bottom-up Model

In the bottom-up model, permissions are generally working as a building block so as to be aggregated into roles. Like the top-down model, this model often uses abstract concepts such as scenarios and business functions in order to both derive and group permissions [9]. In addition, certain attributes of target objects and operations can be used

[1]The difference between role engineering and role administration is often questioned, and their difference is distinct such that the first is more likely an engineering issue of how to analyze and define roles, whereas the second is more likely an administrative issue such as how to manage designed and implemented roles. Role administration is outside of the purview of this study. For more details on role administration, see [17].

for that purpose as well. More specifically speaking, permissions (object and operation pairs) are derived from existing information systems and grouped on the basis of the certain attributes of permissions. The attributes can be drawn from objects, applications, and systems where those permissions are involved. For example, the owner information or ACLs in file objects can be security-relevant attributes and used for grouping permissions as a functional building block of roles. This is usually done through permission derivation process which falls within the scope of system administration. Afterwards, those permissions are aggregated to roles through role composition process which falls within the scope of security administration.

In [19] Thomsen et al. propose an RBAC framework for network enterprises, in which permissions are derived from objects and their methods and roles are derived from the permissions. All these procedures are enabled by an introduction of seven abstract layers for security management: object, object handles, application constraints, application keys, enterprise keys, key chains, and enterprise constraints. The first four layers belong to application developers, who can use their in-depth knowledge of the applications in order to create generic security components. The last three layers are under the control of system administrators, who can use the generic security components as the security building blocks in order to customize the security policy for their organization. They developed *NAPOLEON* tool, which implements the portion of the framework used by the application developer. Subsequently, Epstein and Sandhu propose an approach to leveraging UML language for RE and discuss an exemplary UML modeling case which is based upon the framework proposed by Thomsen et al. [3]. Their approach is straightforward in representing the RBAC framework. However, their approach needs to be improved to address how UML can be used for modeling the process side of RE.

In [9] Neumann and Strembeck present a more concrete approach to engineering roles, which can be derived from business processes, than in [12]. Their approach has two distinct differences in defining an RBAC model from the latter; *scenarios* are used in their approach as a semantic unit for deriving permissions, and in a bottom-up way, permissions obtained from scenarios are aggregated into roles. Their approach can be summarized as follows; usage scenarios are initially identified and modeled, and then permissions are obtained from them. Afterwards, constraints are identified before each of the previously identified scenarios is examined and refined, if necessary. They use the concept of tasks and work profiles for the purpose of grouping the scenarios. Finally, the definition of a concrete RBAC model follows. To use scenarios to capture and elicit system's behaviors and users' needs in requirement engineering phase is a well-known technique exploited in software engineering communities. Scenarios could be expressed, documented, and handled more unambiguously if it is equipped with the syntax and sematics of standards such as the UML language.

2.2.3 *Hybrid Model*

The hybrid model can be described as a mixed approach of top-down and bottom-up models so as to engineer RBAC roles. For example, the role identification process and the permission derivation process in Figure 2 can be conducted in parallel for a later role definition.

In [4], Epstein and Sandhu propose a conceptual framework where roles can be defined in either a top-down or a bottom-up manner. They extend RBAC reference models by introducing three additional layers in between roles and permissions. They use the concepts of jobs, work-patterns, and tasks, to represent those respective layers and facilitate role-permission assignment into smaller and better steps. Top-down and bottom-up models are discussed in conjunction with the notions of *focus* and *bucket*, respectively. However, their work addresses RE in a conceptual manner without discussing how those concepts are specified, constructed, or concretized.

In [8], Kern et al. propose a life cycle model of roles, interwoven with RE and role administration processes. The life cycle model is based on an iterative-incremental process. Four stages of the life cycle of roles are identified: role analysis, role design, role management, and role maintenance. Role analysis is the activity of identifying roles as they occur within the target domain. Role design involves mapping roles onto the system-dependent syntax and semantics as well as designing roles for administration. Role management is the routine role administration such as creation or deletion of a user or a permission and changes in the role model. Role maintenance activities are composed of changes in the mapping of organizational structures to role and changes in the definition of user-role and role-permission relationships.

There seems to be no point in deciding which model is more effective or efficient in RE, because each of them has its own advantages and pitfalls depending upon the varying contexts. In general, we agree with the remark from the authors in [8] that a top-down model is likely to ignore the existing permissions, while a (*pure*) bottom-up model, a model without using any abstract concept to group permissions in our context, is not likely to consider business functions within an organization. In addition, since RE may be the costliest component of RBAC implementation according to a recent study by NIST [5], curtailing the expenses should be a factor in deciding which model to use.

3. UNDERSTANDING SYSTEM-CENTRIC INFORMATION IN RE

In this section we describe our understanding of system-centric information in the context of RE and discuss how it can be used in RE. First, we describe the general characteristics of the information to be elicited for RE, along with the identification of two authoritative boundaries involved in RE. Afterwards, we introduce the two different information flow types between the authoritative boundaries. Then, we discuss the detailed description of how to leverage the system-centric information as a method for both analysis and communication. The design of the information model will be discussed in Section 4.

3.1 Informational Characteristics

Each process to gather, analyze, model, and validate information in RE must be gone through in a complete and accurate manner. This helps prevent possible defects that normally propagate to RBAC design and implementation at a later time. In this section, we identify and describe the general characteristics of the information elicited for RE. We believe that it is necessary to be cognizant of them before discussing system-centric information in detail.

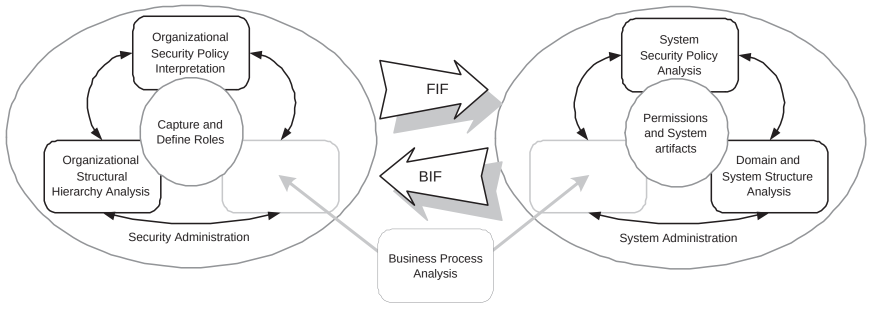

Organizational Security Policy Interpretation
Capture and Define Roles
Organizational Structural Hierarchy Analysis
Security Administration
FIF
BIF
Business Process Analysis
System Security Policy Analysis
Permissions and System artifacts
Domain and System Structure Analysis
System Administration

Figure 3: Information flow types in role engineering

The general characteristics of the information in RE can be described as follows.

- *Different Origins* - The information required for RE, for example, organizational security policies, business process analysis, and system analysis, usually comes from different sources.
- *Distinct Concerns* - The information elicited for RE usually has its own wish-lists in the definition of valid roles. For instance, organizational security policies may concern constraints applied to the roles, while system artifacts may concern permissions on corresponding information systems associated with the roles.
- *Need for Interpretation* - The information elicited for RE may need its interpretation for the interplay among different authorities involved. For example, organization security policies need to be interpreted in order to be applied to specific information systems.
- *Easily Excessive* - The information elicited for RE may exceed a manually manageable size easily, leading to a need for automation or semi-automation methodologies in its management. For example, permissions derived from business process analysis could easily go beyond manual management in quantity.

With respect to the authorities involved in RE, we identify two conceptual boundaries for our purpose, *security administration* and *system administration*[2], which are relevant to two pivotal goals in RE, *role definition* and *permission assignment*, respectively. Our basic assumption for this is that there is no single authority, for example a role engineering group, that is competent to manage and perform all the information and activity required for RE. However, if there is, those boundaries might also be considered to be divisions with distinct expertise under the authority. Figure 3 depicts those boundaries.

3.1.1 Security Administration

The main goal of this boundary pertains to the definition of valid roles within an organizational RBAC environment. Different authorities (or departments) depending upon the organizational structures can be included in this security administration boundary. Three analysis activities are identified in order to accomplish the goal: analysis (and interpretation, if necessary) of organizational security policies, analysis of organizational structural hierarchy, and analysis of business process.

[2]Note that we used the two boundaries in discussing RE models in the previous section for the sake of consistency in this paper.

- *Organizational security policy analysis* - This activity pertains to the analysis of security policies in order to obtain the organization access control policies since access control policies are normally in accordance with security policies. The interpretation of security policies can be considered as a method to apply those policies to specific information systems residing in the system administration boundary.
- *Organizational structural hierarchy analysis* - This analysis is related to the investigation into the organizational structure hierarchy in order to obtain the information which can contribute to possible formulation of preliminary role hierarchies.
- *Business process analysis* - This analysis in the security administration boundary is related to the investigation into business processes in order to derive roles. For instance, we have discussed a top-down RE model where this analysis is leveraged in order to derive roles [12]. Note that this analysis can also be applied in the system administration boundary for a different purpose.

3.1.2 System Administration

The main goal of this boundary concerns facilitating permission identification and assignment. In much the same way as the security administration boundary, this boundary can include different authorities (or departments) partaking in RE process depending upon organizational structures. The goal is carried out by three supportive activities: analysis of system security policies, analysis of domain and system structures, and analysis of business process.

- *System security policy analysis* - This activity pertains to the analysis of the system security policies effective in the existing information systems within an organization. For instance, these security policies could be DAC, MAC, RBAC, and so on.
- *Domain and system structure analysis* - This involves the analysis of both domain structure which are composed of information systems and system structure which can be functionally decomposed into small subsystems. The domain can be analyzed according to either organization units or to network addresses such as DNS.

- *Business process analysis* - This analysis in the system administration boundary pertains to the investigation into business processes with the view of permission derivation. For instance, we have discussed a bottom-up RE model where this analysis is leveraged to derive permissions [9].

3.2 Informational Flows

In RE, the information is circulated from one boundary to another for sharing expertise and knowledge under certain purposes such as role definition and permission assignment. We identify two different informational flow types: *forward information flow* and *backward information flow*. Figure 3 describes the two informational flows.

Forward Information Flow (FIF) In this flow type, the information gathered, analyzed, and validated as a RBAC requirement in the security administration flows into the system administration for the purpose of permission assignment, which can be viewed as a process where the high-level security policies (RBAC) expressed as valid role definitions are manifested into corresponding information systems. Hence, in this flow type, the information can be modeled and used to formalize and communicate the needs of the security administration to the system administration boundary.

Backward Information Flow (BIF) In this flow type, information gathered, analyzed, and validated as a RBAC requirement in the system administration is sent back to the security administration for role definition; the information such as system artifacts or permissions is used to help define the high-level security policies (*valid* roles). The information can be modeled and used to formalize and communicate the expertise of the system administration to the security administration boundary.

3.3 Usages of System-centric Information

As stated earlier, the analysis of business processes generally gives an impetus to deriving RBAC roles (or permissions depending upon which RE model to use), and the knowledge of systems is quite inevitable to provide the clear understanding of both business processes and RBAC roles. The system-centric information, which can be expressed at a much higher level of abstraction than objects and operations, lends itself to two distinct uses as follows.

1. As a method of analysis - A system can be viewed as a single representation of a variety of IT resources. System-centric information provides a groundwork for analyzing IT resources into functionally independent components for the system administration boundary. Its semantics can be more expressive with different types of attributes such as services or states. The analysis has a direct bearing on RBAC permissions derivation or management.

2. As a method of communication - A system can be viewed as a simpler representation decoupling the business processes and RBAC permissions derived from them. Hence, system-centric information, abstracting out the details of object and operation pairs, can provide a groundwork for a good communication of the technical knowledge, at the right level of abstraction, in order to introduce the technical competence to the security administration boundary. The technical competence often leads to role definition in a more concrete manner.

Note that our approach to using the system-centric information does not mean that we completely ignore the notion of objects and operations in the system. Instead, we consider it to be essential in the process of system analysis, since object and operation pairs from the system are grouped into and represented as permissions. We reiterate that considering RE's multi-disciplinary nature, the concept of *system* is more appropriate for the method of analysis and communication in RE than that of object and operation.

4. MODELING SYSTEM-CENTRIC INFORMATION IN RE

An information model which reflects system-level knowledge should accompany well-organized definition and analysis of the significant aspects in systems. This lays a foundation for facilitating the later design and implementation of RBAC. In this section we discuss how to model the system-centric information. We design the information model in such a way that it is particularly appropriate for use in BIF, thereby enhancing consistency in communication of system artifacts and diminishing possible conflicts or misinterpretations between security and system administration boundaries. We use the UML modeling language to convey the syntax and semantics of information systems. We believe that this will benefit in two aspects. Firstly, it can be easy-to-use and reusable. The reusability can be achieved by inheritance. Secondly, we can leverage the existing UML-enabled software tools for modeling system-related knowledge.

4.1 Metamodel

Simplified metamodels are discussed briefly in [12, 16] in order to describe how to implement RBAC in the corresponding organizational environments. Hence, though slightly different in naming, those metamodels generally represent the core concepts in RBAC, and metamodel types such as `Role` and `Access Right` as well as their relationship are defined according to the core concepts.

In Figure 4, a metamodel is shown as an example. Note that we have taken only a part from the metamodel in [12] as the example. The example illustrates four metamodel

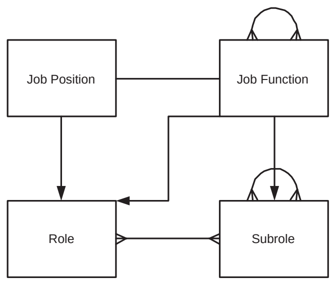


Figure 4: An example of a meta-model for role engineering in the context of security administration - ***Sourced from [12]***

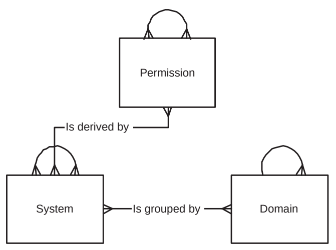


Figure 5: A meta-model for role engineering in the context of system administration

types which formalize the conceptual semantics of roles and their hierarchy. In the example, the relation between `Role` and `Subrole` pertains to role hierarchy, while the relation among `Role` and `Job Position` or `Job Function` shows the semantics of roles, which we discussed earlier. In our context, this exemplary metamodel is considered to be more suitable for use in the security administration boundary.

Figure 5 shows a metamodel reflecting the concepts of system, domain, and permission. In the metamodel, we introduce three new metamodel types, `System`, `Domain`, `Permission` as well as relations between these types. In short, these types are used to formalize the conceptual representation of information systems and permission derivation from the systems in the context of RE. The relation between `System` and `Permission` pertains to the permission derivation. The relation between `System` and `Domain` represents a containment of systems into a domain for the purpose of certain goals such as managing networks or delimiting security. In the following sections, we discuss these metamodel types in more detail by instantiating them into a model.

4.2 Domain Modeling

As shown in Figure 6, we introduce `<<domain>>` stereotype in order to visualize, specify, construct, and document the concept of domain where security and management policies can be applied to information systems in a consistent manner. The semantics of `<<domain>>` stereotype can be both physical and logical due to the flexibility enabled by inheritance relation. Thus, systems can be grouped by both the concept of organizational units or physical network addresses such as DNS. Organizational units can be modeled by means of the UML standard profile for business modeling [10]. This profile defines a stereotype, named `<<Organization Unit>>`, which represents organization units of an enterprise. However, we prefer to using `<<domain>>` stereotype, since it has more appropriate in terms of semantics and terminology. The relation between `<<domain>>` and `<<system>>` is many-to-many association. That is, a domain can consist of zero or more systems. The relation between `<<domain>>` and other `<<domain>>` is one-to-many aggregation. For example, a higher domain may have multiple subordinate domains. This relationship can be viewed as a hierarchical, tree-like structure, not unlike a computer file system structure. Two classes are predefined according to domain hierarchical structure: `Higher` and `Subordinate` classes.

4.3 System Modeling

To visualize, specify, construct, and document the concept of system, we introduce `<<system>>` stereotype. The stereotype of `<<system>>` represents a functional building block of an information system. Figure 6 illustrates the stereotype. The relation between `<<system>>` and `<<rb.permission>>` is many-to-many association. That is, a system can have multiple permissions derived from it, and multiple systems can have one common permission derived from them. The `<<system>>` has to belong to one or more `<<domain>>`. Note that we use the hierarchical domain structure in order for `<<domain>>` to share a common `<<system>>`. The `<<system>>` itself may have many-to-many aggregation relation. For example, an information system can have multiple subsystems, and a subsystem can be aggregated into multiple systems. Four classes are predefined according to functionality: `Collective`, `Process`, `Entity` and `Utility` classes. As for the predefined classes, we borrow the concept of classifying information systems according to their functionality from Herzum and Sims [7]. They propose that information systems can be classified into the following three different types.

Process System This represents business activities in the domain. Examples are Invoice manager and Order Manager.

Entity System This represents the business concepts on which business processes operate. These include both data and repository objects. Examples are invoice, order and products.

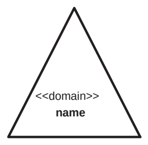
Utility System This is used by both process system and entity system.

4.4 Permission Modeling

We introduce `<<rb.permission>>` stereotype in order to visualize, specify, construct, and document the concept of permission in RBAC. As stated earlier, the relation between `<<rb.permission>>` and `<<system>>` is many-to-many association. Thus its instance must be associated with one or more instance of `<<system>>`. `<<rb.permission>>` itself has many-to-many association relationship. This accounts for the abstract permission and the primitive permission. For instance, an abstract permission can have multiple primitive (basic) permission, and a primitive (basic) permission can be aggregated into multiple abstract permissions. Two classes are predefined according to its granularity: `Abstract` and `Primitive` classes, as shown in Figure 6.

4.5 An Example

We describe an example where our information model can be used for the purpose of demonstrating the feasibility of our approach. We employ the information model in a business process driven framework proposed in [11]. In the framework, simply put, the analysis of business processes defines service components of an application system, called the hospital-based laboratory information system (HLIS). The service components are grouped by information domains. For instance, the patient information domain may comprise patient insurance and patient demographic information. Furthermore, specific objects and operations are derived from the service components. Though the sequential steps for the framework are described briefly, there is little description in conjunction with RE process. Using the information model, we illustrate how to capture and describe the system-centric information for RE under the organizational setting in [11].

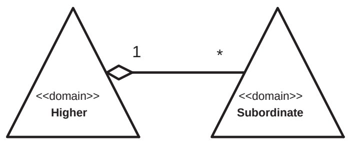


Stereotype: domain

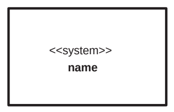
Description
A <<domain>> represents a semantic unit used to accomplish network management goals such as delimiting security or structuring information systems based upon business activities. It comprises multiple <<system>>.
Base class
Class
TagDefinition

property[*]:	String
description :	String

Constraint
1. A <<domain>> must be associated with zero or more instance of <<system>>.
2. A <<domain>> itself has one-to-many association.
Predefined classes
Higher Domain, Subordinate Domain



Predefined classes

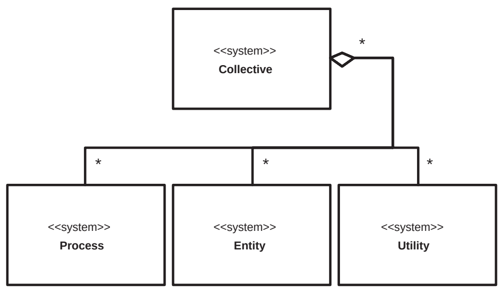


Stereotype: system

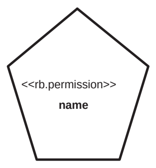
Description
A <<system>> represents a semantic unit pertaining to a functional building block of an information system.
Base class
Class
TagDefinition

authority:	<<domain>>
property[*]:	String
description :	String

Constraint
1. A <<system>> must be associated with one or more instance of <<permission>>.
2. A <<system>> must be associated with one or more instance of <<domain>>
2. A <<system>> itself has many-to-many association.
Predefined classes
Collective System, Process System, Entity System, Utility System.



Predefined classes

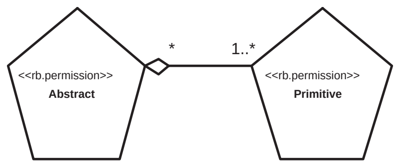


Stereotype: rb.permission

Description
A <<rb.permission>> represents a semantic unit pertaining to a particular mode of access to one or more <<system>>. It provides a functional building block of a role.
Base class
Class
TagDefinition

object:	String
operation:	String
property[*]:	String
description:	String

Constraint
1. A <<rb.permission>> must be associated with one or more instance of <<system>>.
2. A <<rb.permission>> itself has many-to-many association.
Predefined classes
Abstract Permission, Primitive Permission



Predefined classes

Figure 6: Domain, system, and permission stereotype

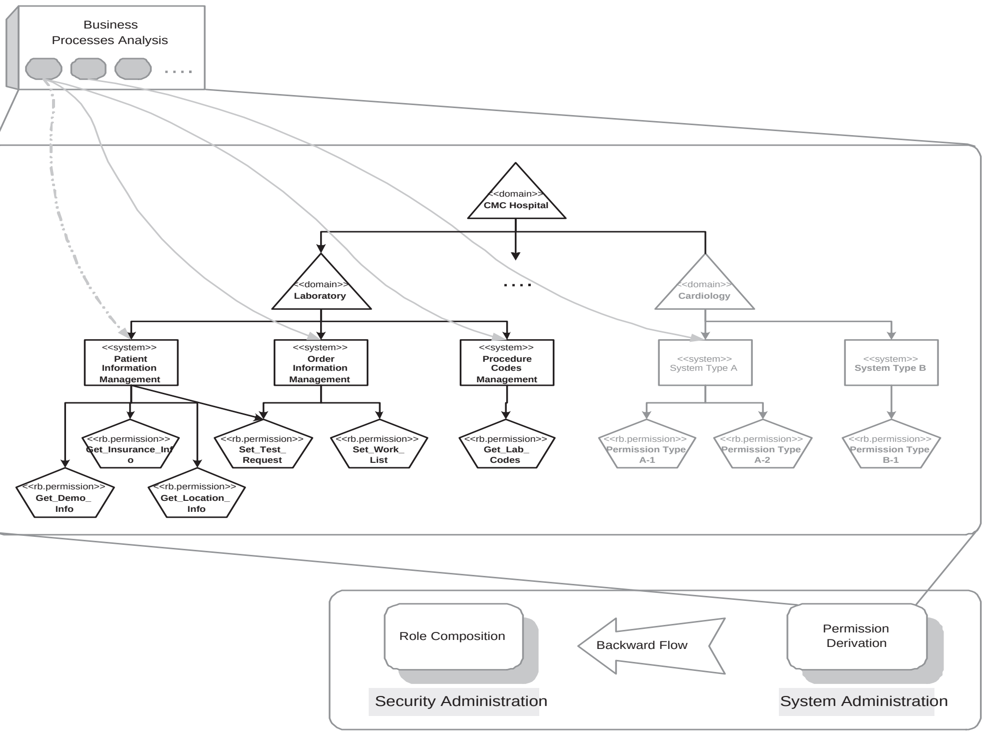

Business Processes Analysis
....
<<domain>> CMC Hospital
<<domain>> Laboratory
....
<<domain>> Cardiology
<<system>> Patient Information Management
<<system>> Order Information Management
<<system>> Procedure Codes Management
<<system>> System Type A
<<system>> System Type B
<<rb.permission>> Get_Insurance_Info
<<rb.permission>> Set_Test_ Request
<<rb.permission>> Set_Work_ List
<<rb.permission>> Get_Lab_ Codes
<<rb.permission>> Permission Type A-1
<<rb.permission>> Permission Type A-2
<<rb.permission>> Permission Type B-1
<<rb.permission>> Get_Demo_ Info
<<rb.permission>> Get_Location_ Info
Role Composition
Backward Flow
Permission Derivation
Security Administration
System Administration

Figure 7: Domain, system, and permission diagram

Figure 7 shows the domain, system, and permission diagram with a graphical and concise view of system-centric information of the exemplary organizational setting. Since the bottom-up model is used, the analysis of business processes takes place within the system administration boundary in order to derive system-centric information in general and permissions in specific. Permissions as object and operation pairs within systems are drawn on the basis of the analysis of business processes. For example, assuming that a process of lab order entry is identified and analyzed under the laboratory domain, it concerns three information systems to accomplish the business process: *patient information management*, *order information management*, and *procedures code management*. All of the three information systems are collective systems comprising process systems (application logic) and entity systems (database). Afterwards, six permissions are derived from the identified systems: *Get_Demo_Info*, *Get_Insurance_Info*, *Get_Location_Info*, *Set_Test_Request*, *Set_Work_List*, and *Get_Lab_Codes*. All of these permissions are abstract permissions, which consist of one or more primitive permissions, for instance, pairs of object (*patient* table) and operation (*select* statement). This documented information flows back to the security administration boundary to help define roles in a more concrete manner on the basis of the knowledge of domain, system, and permission.

5. DISCUSSION AND CONCLUSION

Role engineering may be costly and time-consuming in that large amounts of information need to be exchanged, collected and so much activity is required of different authorities involved. In this paper, we discuss an approach to modeling system-centric information, which can be expressed both at a much higher level of abstraction and with fairly less amount of information than that of objects and operations, with a view to facilitating RE. We first describe the general informational characteristics in RE. Afterwards, we discuss two types of information flow among authorities in RE process, *forward information flow (FIF)* and *backward information flow (BIF)*. Then, we introduce an information model which is greatly useful in BIF. System-centric information is incorporated in the information model and UML extension mechanisms are exploited for modeling those information. The information model works both as a method to analyze system-centric information and as an effective vehicle for communication system-related knowledge. An exemplary use of the information model is described to demonstrate its feasibility.

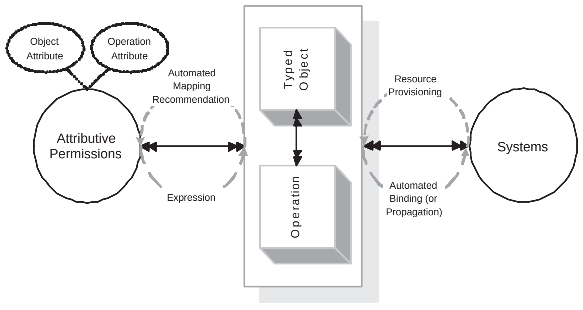


Figure 8: Permission derivation using the concept of attributive permission

Our future work will be conducted in both horizontal and vertical directions for the purpose of extending the information model. As a horizontal, we will investigate an information model which is suitable for use in the security administration boundary. As a vertical, we will explore how to model the business process in UML so that it can be expressed in conjunction with the proposed information model as a whole in RE. In addition, further specifications of the proposed information model which can account for a variety of dynamic and static aspects of system-centric information will be also explored. From the perspective of the implementation in RE, an approach to deriving or managing permissions in automation is certainly an interesting topic. Figure 8 shows the concept of attributive permission, which may be used for permission derivation in such a way. The basic concept of the attributive permission concerns two types of attributes which a permission can have for the purpose of automation in the permission derivation: object attribute and operation attribute. Those attribute are expressed by a role engineer in order to obtain recommended permissions in a automatic way. Our future work will investigate how to concretize the concept of attributive permission.

6. ACKNOWLEDGMENT

This work was partially supported at the Laboratory of Information of Integration, Security and Privacy at the University of North Carolina at Charlotte by the grants from National Science Foundation (NSF-CCR-0124873) and the Electronics and Telecommunications Research Institute.

7. REFERENCES

[1] G.-J. Ahn and R. Sandhu. Role-based authorization constraints specification. *ACM Transactions on Information and System Security*, 3(4), November 2000.

[2] E. Coyne. Role engineering. In *Proceedings of 1st ACM Workshop on Role-Based Access Control*, Gaithersburg, MD, November 1995.

[3] P. Epstein and R. Sandhu. Towards a UML based approach to role engineering. In *Proceedings of 4th ACM Workshop on Role-Based Access Control*, pages 33–42, Fairfax, VA, October 28-29 1999.

[4] P. Epstein and R. Sandhu. Engineering of role/permission assignment. In *Proceedings of 17th Annual Computer Security Application Conference*, New Orleans, LA, December 2001.

[5] M. P. Gallaher, A. C. O'Connor, and B. Kropp. The economic impact of role-based access control. Planning report 02-1, National Institute of Standards and Technology, March 2002.

[6] C. Goh and A. Baldwin. Towards a more complete model for role. In *Proceedings of 3rd ACM Workshop on Role-Based Access Control*, Fairfax, VA, October 22-23 1998.

[7] P. Herzum and O. Sims. The business component approach. In *OOPSLA'98 Business Object Workshop IV*, Vancouver, Canada, July 13 1998.

[8] A. Kern, M. Kuhlmann, A. Schaad, and J. Moffett. Observations on the role life-cycle in the context of enterprise security management. In *Proceedings of 7th ACM Symposium on Access Control Models and Technologies*, Monterey, CA, June 2002.

[9] G. Neumann and M. Strembeck. A scenario-driven role engineering process for functional RBAC roles. In *Proceedings of 7th ACM Symposium on Access Control Models and Technologies*, Monterey, CA, June 2002.

[10] OMG. Unified modeling language UML specification v1.4. Technical report, September 2001.

[11] C. Ramaswamy. Business process driven framework for defining an access control service based on roles and rules. In *23rd National Information Systems Security Conference*, Baltimore, MD, October 16-19 2000.

[12] H. Roeckle, G. Schimpf, and R. Weidinger. Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. In *Proceedings of 5th ACM Workshop on Role-Based Access Control*, Berlin, Germany, July 26-27 2000.

[13] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security*, 2(1), February 1999.

[14] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[15] R. S. Sandhu, D. Ferraiolo, and D. Kuhn. The NIST model for role-based access control: Towards a unified standard. In *Proceedings of 5th ACM Workshop on Role-Based Access Control*, Berlin, Germany, July 26-27 2000.

[16] A. Schaad, J. Moffett, and J. Jacob. The role-based access control system of a european bank: A case study and discussion. In *Proceedings of 6th ACM Symposium on Access Control Models and Technologies*, Chantilly, VA, May 3-4 2001.

[17] D. Shin, G.-J. Ahn, S. Cho, and S. Jin. A role administration system in role-based authorization infrastructures - design and implementation. In *Proceedings of 18th ACM Symposium on Applied Computing*, Melbourne, FL, March 9-12 2003.

[18] Z. Tari and S.-W. Chan. A role-based access control for intranet security. *IEEE Internet Computing*, September-October 1997.

[19] D. Thomsen, D. O'Brien, and J. Bogle. Role based access control framework for network enterprises. In *Proceedings of 14th Annual Computer Security Application Conference*, pages 50–58, Scotsdale, AZ, December 7-11 1998.

[20] L. Zhang, G.-J. Ahn, and B. Chu. A rule-based framework for role-based delegation. In *Proceedings of 6th ACM Symposium on Access Control Models and Technologies*, pages 153–162, Chantilly, VA, May 3-4 2001.