A Formal Framework to Elicit Roles with Business Meaning in RBAC Systems

Alessandro Colantonio
Engiweb Security
Roma, Italy
alessandro.colantonio@eng.it

Roberto Di Pietro
Università di Roma Tre
Roma, Italy
dipietro@mat.uniroma3.it

Alberto Ocello
Engiweb Security
Roma, Italy
alberto.ocello@eng.it

Nino Vincenzo Verde
Università di Roma Tre
Roma, Italy
nverde@mat.uniroma3.it

ABSTRACT

The *role-based access control* (RBAC) model has proven to be cost effective to reduce the complexity and costs of access permission management. To maximize the advantages offered by RBAC, the *role engineering* discipline has been introduced. A viable approach is to explore current applications and systems to find de facto roles embedded in existing user permissions, leading to what is usually referred to as *role mining*. However, a key problem that has not yet been adequately addressed by existing role mining approaches is how to propose roles that have *business meaning*. In order to do this, we provide a new formal framework that also enjoys practical relevance. In particular, the proposed framework leverages business information—such as business processes and organization structure—to implement role mining algorithms. Our key observation is that a role is likely to be meaningful from a business perspective when it involves activities within the same business process or organizational units within the same branch. To measure the "spreading" of a role among business processes or organization structure, we resort to *centrality* indices. Such indices are used in our cost-driven approach during the role mining process. Finally, we illustrate the application of the framework through a few examples.

Categories and Subject Descriptors

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

General Terms

Theory, Experimentation, Measurement

Keywords

Role-Based Access Control, RBAC, Role Engineering, Role Mining, Business Meaning

SACMAT'09, June 3–5, 2009, Stresa, Italy.
ACM 978-1-60558-537-6/09/06.

1. INTRODUCTION

Among all proposed access control models in the current literature, *role-based access control* (RBAC, [1]) is certainly the most adopted by medium- to large-size organizations, greatly due to the simplicity of the model. The most important benefit of adopting RBAC is that *it minimizes system administration effort* due to the reduced number of relationships required to relate users to permissions [2,17]. Yet, RBAC offers benefits to business users as well. A role represents a job function or a title established for a set of users within an organization. Thus, the adoption of RBAC makes it easier to define security policies by business users who have no knowledge of IT systems. For this reason, *business alignment* is always required during the security policies definition and, in particular, to establish the rational for roles.

Although the adoption of RBAC introduces benefits from both a system and business perspective, many organizations are still reluctant to deploy role-based access control systems, since there are still some important issues that need to be addressed. In particular, to maximize the advantages offered by roles, the model must be customized to capture the needs and functions of the company. For this purpose, the *role engineering* discipline [6] has been introduced. However, identifying roles from scratch is a complex task. Choosing the best way to design a proper set of roles is still an open problem. Various approaches have currently been proposed in order to solve it. These are usually classified as: *top-down* and *bottom-up*. The former requires a deep analysis of business processes to identify which access permissions are necessary to carry out specific tasks. The latter seeks to identify de facto roles embedded in existing access permissions resorting to data mining techniques, thus leading to what is usually referred to as *role mining*. Both top-down and bottom-up approaches have pros and cons. Top-down may ignore existing permissions and exceptions, whereas bottom-up may not consider business functions of an organization [13]. For this reason, bottom-up may be used in conjunction with top-down, leading to an *hybrid* approach.

Since the bottom-up role engineering approach can be automated, it has attracted a lot of interest from researchers. Nevertheless, a key problem that remains is how to propose roles that have *business meaning*. Roles discovered by analyzing existing access permissions through role mining algorithms are often no more than a set of permissions with no

connection to the business practice. Indeed, the main objective of most of the role mining approaches is only to reduce the number of roles or to simplify the access control administration from a system perspective. But organizations are unwilling to deploy roles they cannot understand, even though such roles are limited in number. Only a few recent works value business requirements in role mining [2, 17]. However, it is still unclear how to model business-related data so as to elicit roles that could be interpreted in business terms also.

Contributions. This paper provides a new formal framework for role engineering that also enjoys practical relevance. The framework allows for the implementation of a concrete hybrid role engineering approach through the combination of existing role mining algorithms (bottom-up approach) and business analysis (top-down approach). In particular, we describe how business processes and organization structure can be modeled in order to define a *metric* for evaluating the business meaning of candidate role-sets. This metric is used during the role mining algorithm execution to establish which roles should be included in the candidate role-set. The key insight is that a role is likely to be meaningful from a business perspective when: it involves activities within the same business process; or, it involves users belonging to the same organization structure branch. To measure the "spreading" of a role among business processes or organization units we resort to *centrality*, a well-known concept in graph theory. In particular, we define two different centrality indices, namely the *activity-spread* and the *organization-unit-spread*. Leveraging our cost-driven approach [2] makes it feasible to take into account the proposed indices during the role mining process. Finally, we demonstrate the effectiveness of our proposal through a few examples and applications to real data.

Roadmap. This paper is organized as follows: Section 2 reports on related works. Section 3 introduces the background required to formally describe the framework. Section 4 describes the proposed framework by introducing the activity-spread and the organization-unit-spread indices. Section 5 illustrates an example of application of the framework and related discussion. Finally, Section 6 provides concluding remarks.

2. RELATED WORK

Role engineering was first illustrated by Coyne [6], providing a top-down perspective. He places system users' activities as high-level information for role identification; this approach is only conceptual, thus it lacks technical details. Fernandez and Hawkins [11] propose a similar approach where use-cases are used to determine the needed permissions. Röckle et al. [22] propose a process-oriented approach that analyzes business processes to deduce roles. Crook et al. [7] leverage organizational theory to elicit role-based security policies. Neumann and Strembeck [18] offer a scenario-based approach where a usage scenario is the basic semantic unit to analyze. Work-patterns involving roles are analyzed and decomposed into smaller units. Such smaller units are consequently mapped with system permissions. Shin et al. [24] use a system-centric approach supported by the UML language to conduct top-down role engineering. Epstein and Sandhu [10] also use UML to address role engineering. Kern et al. [13] propose an iterative and incremental approach based on the role life-cycle, pertaining to analysis, design, management, and maintenance.

All the abovementioned works represent pure top-down approaches, namely they do not consider existing access permissions; hence, they do not take into account how the organization actually works. As for the bottom-up approach, Kuhlmann et al. [14] first introduced the term "role mining", trying to apply existing data mining techniques to elicit roles from existing access data. The first algorithm explicitly designed for role engineering is ORCA [23] which applies hierarchical clustering techniques on permissions. However, this approach does not allow for permission overlapping among roles that are not hierarchically related. Vaidya at al. [26] applied subset enumeration techniques to generate a set of candidate roles, computing all possible intersections among permissions possessed by users. More recently, they also studied the problem of finding the minimum number of roles that cover all permissions possessed by users [25]. By leveraging binary integer programming, Lu et al. [16] presented a unified framework for modeling the role number minimization problem. Ene et al. [9] offered yet another alternative model to minimize the number of roles, reducing it to the well-known problem of the minimum biclique covering. Zhang et al. [28] provide an attempt to contextually minimize the number of user-role, permission-role, and role-role relationships. Frank et al. [12] model the probability of user-permission relationships, thus allowing to infer the role-user and role-permission assignments so that the direct assignments become more likely.

The main limitation of all the cited role mining approaches is that they do not always lead to the optimal set of roles from a business perspective. To the best of our knowledge, the work from Colantonio et al. [2] represents the first approach that allows for the discovery of roles with business meanings through a role mining algorithm. A cost function is introduced as a metric for evaluating "good" collections of roles. Minimizing the cost function makes it possible to elicit those roles that contextually minimize the overall administration effort and fit the needs of an organization from a business perspective. Further improvements of this work are represented by [3–5]. The most similar approach to [2] is later provided by Molloy at al. [17]. Molloy et al. utilize user attributes to provide a measurement of the RBAC state complexity, called "weighted structural complexity". This measure is then used with an algorithm based on formal concept analysis (FCA) to reduce the complexity of the discovered roles.

3. BACKGROUND AND PRELIMINARIES

In this section we introduce some concepts and tools required to formally describe our framework. We first summarize the RBAC model (Section 3.1) and the cost-driven approach (Section 3.2) which represent the foundation of our approach. Activities (Section 3.3) and organization units (Section 3.4) are provided as distinguishing elements of the framework. Then, the farness index (Section 3.5) is introduced to evaluate the business meaning of roles.

3.1 Role-Based Access Control

We shall now review some concepts of the ANSI/INCITS RBAC standard [1] required for the present analysis. In particular, we are interested in the following ones:

- *PERMS*, the set of all possible access permissions;
- *USERS*, the set of all system users;
- *ROLES*, the set of all roles;
- $UA \subseteq USERS \times ROLES$, the set of all role-user relationships;
- $PA \subseteq PERMS \times ROLES$, the set of all role-permission relationships;
- $RH \subseteq ROLES \times ROLES$, the set of hierarchical relationships between roles.

RH derives from the partial order based on permission-set inclusion.[1] The symbol '$\succeq$' indicates the ordering operator. If $r_1 \succeq r_2$, then r_1 is referred to as the *senior* of r_2, namely r_1 adds certain permissions to those of r_2. Conversely, r_2 is the *junior* of r_1. The following functions are also provided:

- *assigned_users*: $ROLES \rightarrow 2^{USERS}$ to identify users assigned to a role and to none of its senior roles.[2]
- *authorized_users*: $ROLES \rightarrow 2^{USERS}$ to identify users assigned to a role or to at least one of its seniors.
- *assigned_perms*: $ROLES \rightarrow 2^{PERMS}$ to identify permissions assigned to a role and to none of its senior roles.[3]
- *authorized_perms*: $ROLES \rightarrow 2^{PERMS}$ to identify permissions assigned to a role or to at least one of its seniors.

In this paper we also introduce $UP \subseteq USERS \times PERMS$ as the set of user-permission assignments to be analyzed.

3.2 Cost-Driven Approach

This section formally describes the cost-driven approach. According to [2, 5], candidate roles are identified based on the measurement and evaluation of their introduced advantages during the entire role mining process. By adopting the *cost function* concept it is possible to elicit roles that minimize the related administration cost. A cost function is a combination of several *cost elements*, each of them considering a particular business- or IT-related aspect. Among the data available to the organization, it is possible to find information that might either directly influence the required system administration effort (e.g., number of roles, number of role-user relationships to be administered, etc.) or information that might help role engineers assign business meaning to roles (e.g., business processes, organization structure, etc.). Once an organization has identified the relevant data for access control purposes, this data should be "translated" into cost elements and then combined into a cost function. This makes it possible to identify the *optimal* candidate roles which best describes the actual needs of the organization. More specifically, minimizing the cost function can simultaneously optimize the administration effort as well as the business meaning related to the elicited roles.

The approach is founded on the following definitions.

[1]The RBAC papers that mention role hierarchy most often treat it as a partial order. Since consensus (on this matter) has yet to be reached among researchers [15], we only consider hierarchical relationships derived from permission-set inclusion.

[2]The RBAC standard does not make a clear distinction between base and derived relations [15]. We therefore consider the functions *assigned_users* as derived from *UA*, that is $assigned\_users(r) = \{u \in USERS \mid \langle u, r\rangle \in UA\}$. We also assume that users are assigned to a role are not assigned to its seniors.

[3]Analogous to *assigned_users*, we consider the function *assigned_perms* as derived from *PA*, that is $assigned\_perms(r) = \{p \in PERMS \mid \langle p, r\rangle \in PA\}$. We also assume that permissions assigned to a role are not assigned to its juniors.

DEFINITION 1 (SYSTEM CONFIGURATION). *Given an access control system, we refer to its* configuration *as the tuple* $\varphi = \langle USERS, PERMS, UP\rangle$*, that is the set of existing users, permissions, and the relationships between them.*

DEFINITION 2 (RBAC STATE). *Given an RBAC system, we refer to its* state *as* $\gamma = \langle ROLES, UA, PA, RH\rangle$*, namely an instance of the RBAC model.*

DEFINITION 3 (CANDIDATE RBAC STATE). *Given the system configuration* $\varphi = \langle USERS, PERMS, UP\rangle$*, a* candidate RBAC state $\gamma = \langle ROLES, UA, PA, RH\rangle$ *is a state where its roles "cover" all possible combinations of permissions possessed by users according to* φ*, namely*

$$\forall u \in USERS,\ \exists R \subseteq ROLES : \bigcup_{r \in R} authorized\_perms(r) = \{p \in PERMS \mid \langle u, p\rangle \in UP\}.$$

DEFINITION 4 (COST FUNCTION). *Given the system configuration* $\varphi = \langle USERS, PERMS, UP\rangle$*, let* Γ_φ *be the space of all possibile candidate RBAC states for* φ*. We refer to the* cost function *as*

$$c\colon \Gamma_\varphi \rightarrow \mathbb{R}.$$

It estimates the administration effort for an RBAC state.

Definition 4 shows the simplest form of a cost function. However, a more complex function may include in its domain other elements which influence the administration cost.

DEFINITION 5 (OPTIMAL CANDIDATE RBAC STATE). *Given a configuration* φ *and a cost function* c*, an* optimal candidate RBAC state γ *is a candidate RBAC state for* φ *that minimizes the function* c*.*

3.3 Activities

Activities (or *tasks*) are a natural way to think about user actions and their contexts. Activities usually arise from the decomposition of the *business processes* of an organization. A business process is a collection of inter-related activities that accomplish a specific goal. From an access control point of view, an activity induces a set of permissions necessary to perform an elementary part of a more complex job. Activities are typically assigned to users on the basis of their job positions [19].

Since the activity and the role concepts are similar in that they both group permissions, one might think that there is no difference between them. However, they have completely different meanings and characteristics. A role focuses on "actors" and places emphasis on *how* the business should be organized, while an activity focuses on "actions" and emphasizes *what* should be done. For example, a typical workflow management system requires that each actor should complete certain tasks. In this case, each task may be performed by several actors with different business roles. It would be ineffective to assign an RBAC role to each task. Neither should an activity be considered a "sub-role": role-permission relationships are identified with different rationales from activity-permission relationship identification.

It is important to highlight that defining and maintaining the activity model up to date can increase the workload of business users within the company. However, a good understanding of the organization is a mandatory requirement

when implementing an access control management framework. Therefore, the activity model is already available within an organization.

In the following we formally describe the activity concept.

Activity Tree. Decomposing the business processes of an organization usually results in an activity tree structure. For the sake of simplicity, we do not make a formal distinction between business processes and activities. In particular:

- The set $ACTVT$ contains all activities.
- The set $ACTVT\text{-}H \subseteq ACTVT \times ACTVT \times \mathbb{R}$ defines a partial order on the hierarchy tree. The pair $\langle a_p, a_c, w\rangle \in ACTVT\text{-}H$ indicates that the activity a_p is the parent of the activity a_c, whereas w is the *weight* of the connection between a_p and a_c (see below). The existence of the tuple $\langle a_p, a_c, w\rangle$ in $ACTVT\text{-}H$ may alternatively be indicated as $a_c \xrightarrow{w} a_p$. The simplified notation $a_c \rightarrow a_p$ can also be used when w is always 1.
- $\forall a \in ACTVT$, the activity a has only one direct parent, namely $\forall a_p, a_c \in ACTVT : a_c \rightarrow a_p \implies \nexists a'_p \in ACTVT : a'_p \neq a_p, a_c \rightarrow a'_p$.
- The activity tree has only one root.

Given a pair $a_p, a_c \in ACTVT$, the ordering operator $a_c \succeq a_p$ indicates the existence of a hierarchical pathway of "$\rightarrow$" from a_c to a_p. Note that, without loss of generality, it is always possible to identify a unique root for a set of activities; for example, a virtual activity that "collects" all the high level activities of the organization can always be defined.

Given an activity $a \in ACTVT$, the following sets can be defined out of convenience:

- $\uparrow a = \{a' \in ACTVT \mid a \succeq a'\}$ represents all possible parents of a. Since each activity has only one direct parent in the tree, $\uparrow a$ contains the path from a to the root of $ACTVT\text{-}H$. Note that $|\uparrow a|$ is the length of this path. Given a pair $a_1, a_2 \in ACTVT$, the value of $|\uparrow a_1 \cap \uparrow a_2|$ represents the length of the path from the root to the "nearest" common parent of both a_1, a_2.
- $\downarrow a = \{a' \in ACTVT \mid a' \succeq a\}$ represents all possible children of a.

Each activity is supported by sets of permissions which allow the activity to be performed. To execute a given activity, a user must have *all* the permissions associated to it. This concept can be formalized as follows:

- The set $ACTVT\text{-}A \subseteq ACTVT \times PERMS$ expresses the origin of a permission in a given activity.
- The function $actvt\_perms$: $ACTVT \rightarrow 2^{PERMS}$ provides the set of permissions associated to an activity. Given $a \in ACTVT$, it can be formalized as: $actvt\_perms(a) = \{p \in PERMS \mid \exists \langle a, p\rangle \in ACTVT\text{-}A\}$.
- The function $actvt\_perms^*$: $ACTVT \rightarrow 2^{PERMS}$ provides all the permissions assigned to a and its children, namely it takes into account the activity breakdown structure. Given $a \in ACTVT$, it can be formalized as: $actvt\_perms^*(a) = \{p \in PERMS \mid \exists a' \in \downarrow a : \langle a', p\rangle \in ACTVT\text{-}A\}$.

A permission can belong to multiple activities. Moreover, given the activity pair $a_1, a_2 \in ACTVT$ such that $a_1 \succeq a_2$ and $p \in PERMS$, if $\langle p, a_1\rangle \in ACTVT\text{-}A$ then we require that $\langle p, a_2\rangle \notin ACTVT\text{-}A$ since a_2 inherits p from its child.

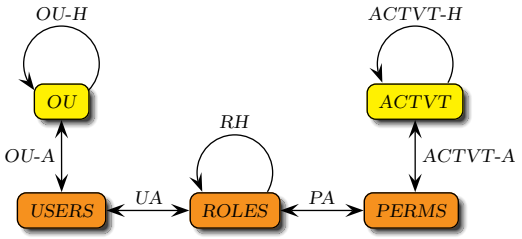

Figure 1: Relationships among activities, organization units and the standard RBAC entities.

Connection Weights. Assigning a weight to connections between activities is a flexible way to model the business process break-down structure. In particular, weights indicate if there is a strong or a weak decomposition. For example, there is likely to be a large weight value between the root activity and activities such as "Human Resources Management" or "Inventory Management". Conversely, the weight between the parent activity "Customer Data Management" and its child "Customer Data Update" is likely to be weak. How weights are derived depends on the given organization, however this topic is not further analyzed in this paper.

The given weight definition can easily be extended to the partial order as the sum of weights along the path between activities. Given $a_c, a_p \in ACTVT : a_c \succeq a_p$, the weight of the path between them is:

$$w_{\text{actvt}}(a_c \succeq a_p) = \textstyle\sum_{\omega \in \Omega} \omega,\ \Omega = \{\omega \in \mathbb{R} \mid \forall a, a' \in ACTVT, \exists \langle a, a', \omega\rangle \in ACTVT\text{-}H : a_c \succeq a',\ a' \succeq a_p\}. \quad (1)$$

Figure 1 gives a graphical representation of the abovementioned entities and the interaction with the RBAC entities. It also depicts other elements described in next section.

3.4 Organization Units

An *organization unit* (OU) is a group of employees which collaborate to perform certain business tasks. The organizational structure is designed by the top management and defines the lines of authority and the division of work. A typical organization is organized as a tree structure, represented by an organization chart. From an access control point of view, it is likely that users of the same OU have the same access permissions. For this reason, users usually have roles located within the organization units [21].

There are many examples of benefits related to the introduction of the organization structures into the access control model. For instance, OUs are used in various frameworks as a means to identify user pools [20, 21]. In some of these works, roles can be assigned directly with OUs instead of individually with users. In this paper, we prefer to directly assign users to roles, thus allowing a complete delegation of organization unit maintenance to HR.

The following is a formal description of the organization unit structure, that reflects the notation used in Section 3.3.

Organization Unit Tree. Organization units can usually be represented in a tree structure. In particular:

- The set OU contains all organization units.
- The set $OU\text{-}H \subseteq OU \times OU \times \mathbb{R}$ defines a partial order on the hierarchy tree. The pair $\langle o_p, o_c, w\rangle \in OU\text{-}H$ indicates that the organization unit o_p is the parent

of the organization unit o_c, whereas w is the *weight* of the connection between o_p and o_c (see below). The existence of the tuple $\langle o_p, o_c, w\rangle$ in *OU-H* may alternatively be indicated as $o_c \xrightarrow{w} o_p$. The simplified notation $o_c \rightarrow o_p$ can also be used when w is always 1.
- $\forall a \in OU$, the organization unit a has only one direct parent, namely $\forall o_p, o_c \in OU : o_c \rightarrow o_p \implies \nexists a'_p \in OU : a'_p \neq o_p, o_c \rightarrow a'_p$.
- The organization unit tree has only one root.

Given a pair $o_p, o_c \in OU$, the ordering operator $o_c \succeq o_p$ indicates the existence of a hierarchical pathway of "$\rightarrow$" from o_c to o_p. Note that, without loss of generality, it is always possible to identify a unique root for the organization units—namely, a unit representing the entire organization. Given an organization unit $o \in OU$, we define:

- $\uparrow o = \{o' \in OU \mid o \succeq o'\}$ represents all possible parents of o. It contains the path from o to the root of *OU-H*, thus $|\uparrow o|$ is the length of this path. Given $o_1, o_2 \in OU$, $|\uparrow o_1 \cap \uparrow o_2|$ is the length of the path from the root to the "nearest" common parent of both o_1, o_2.
- $\downarrow o = \{o' \in OU \mid o' \succeq o\}$ is the set of all children of o.

Each organization unit contains a sets of users. This concept can be formalized as follows:

- The set $OU\text{-}A \subseteq OU \times USERS$ expresses the origin of a user in a given organization unit.
- The function $ou\_users\colon OU \rightarrow 2^{USERS}$ provides the set of users belonging to an organization unit. Given an organization unit $o \in OU$, it can be formalized as: $ou\_users(o) = \{u \in USERS \mid \exists \langle o, u\rangle \in OU\text{-}A\}$.
- The function $ou\_users^*\colon OU \rightarrow 2^{USERS}$ provides all the users belonging to o and its children, namely it takes into account the organization unit breakdown structure. Given $o \in OU$, it can be formalized as: $ou\_users^*(o) = \{u \in USERS \mid \exists o' \in \downarrow o : \langle o', u\rangle \in OU\text{-}A\}$.

Usually, each user belongs to only one organization unit. Hence, given $o_1, o_2 \in OU$ and $u \in USERS$, if $\langle u, o_1\rangle \in OU\text{-}A$, then $\langle u, o_2\rangle \notin OU\text{-}A$.

Connection Weights. Weighting connections between OUs is a new concept when compared to the existing framework related to OUs. It is a more flexible way to model the organization break-down structure. For example, user sets can be divided into various *administrative domains* represented by organization unit branches [20]. It is assumed that each domain is independently administered. In such a case, weights may indicate whether domains are loosely or tightly coupled. Another case is when OUs are decomposed into a set of branches to model geographic areas; this often represents a weak partitioning since there are no big differences among users across different geographic areas. Decomposing a project in various working teams is another example of weak partitioning, since all users work for the same objectives. Conversely, domains represented by business units are more important as they usually identify users assigned with completely different jobs. Therefore, this is an example of strong OU partitioning.

The weight concept can be easily extended to the partial order as the sum of all weights between units along the shortest path between them. Given $o_c, o_p \in OU : o_c \succeq o_p$, the weight of the path between them is:

$$w_{\text{ou}}(o_c \succeq o_p) = \textstyle\sum_{\omega\in\Omega} \omega,\ \Omega = \{\omega \in \mathbb{R} \mid \forall o, o' \in OU, \exists \langle o, o', \omega\rangle \in OU\text{-}H : o_c \succeq o',\ o \succeq o_p\}. \quad (2)$$

Figure 1 gives a representation of the abovementioned entities and the interaction with standard RBAC entities.

3.5 Farness

We now introduce a tool that is particularly useful in topology and related areas in mathematics. This tool will be adapted to our analysis in order to consider business information during the role mining process. Given a graph, we usually refer to a sequence of vertices connected by edges as a *walk*. We also refer to a walk with no repeated vertices as a *path*, while the shortest path between two vertices is referred to as a *geodesic*. These concepts make it possible to introduce the *farness* index, namely a quantity related to the lengths of the geodesics from one vertex to every other vertex in the graph. Vertices that "tend" to have long geodesic distances to other vertices within the graph have higher farness. In the literature it is more common to find another index in place of farness, that is the *closeness* index. Closeness is the inverse of farness, so they can be considered perfectly equivalent. Both farness and closeness are examples of *centrality* measures [27].

The farness index that is used in this paper can be defined in every metric space where a notion of distance between elements is defined. Given a vertex v_j, its farness f is

$$f = \frac{\sum_{i=1}^{n} d(i,j)}{n-1}, \quad (3)$$

where $d(i,j)$ is the distance between the vertices v_i and v_j.

Other centrality measures such as *betweenness*, *degree*, and *eigenvector* centrality [27] might be applicable to our analysis. Since farness is the most suitable and simple one for our analysis, we omit discussions of the others.

4. FRAMEWORK DESCRIPTION

In this section we demonstrate how the business processes model (see Section 3.3) and the organization structure model (see Section 3.4) can be leveraged to evaluate the business meaning of roles elicited by role mining algorithms. The key intuition is that a role is likely to be meaningful from a business perspective when: it involves activities within the same business process; or, it involves users belonging to the same organization structure branch—a justification for this can be found in the following sections. To measure the business meaning of roles, we introduce the following indices:

- *activity-spread* (detailed in Section 4.1) that measures the "dispersion" of business activities that are enabled by permissions assigned to a role;
- *organization-unit-spread* (detailed in Section 4.2) that measures the "dispersion" of organization units that users assigned to roles belong to.

The reason why we resort to business processes and organization structure is quite simple. Given $r \in ROLES$, it identifies both a set of permissions ($authorized\_perms(r)$) and a set of users ($authorized\_users(r)$). Hence, both these sets should be analyzed in order to evaluate the "quality" of the candidate role. Regarding the user set, the structure of

the organization is probably the most significant business-related information that is always available in every medium to large sized organization. As a matter of fact, it is usually found within the HR-related IT systems. Similarly, business activities represent the main justification for the adoption of IT applications within a company. Indeed, each application is usually introduced to support business activities. Usually, the business activity tree can be provided by business staff.

Once activity-spread and organization-unit-spread indices are defined, we propose to adopt our cost-driven approach (see Section 3.2) to take them into account during the role mining process. In particular, leveraging the cost function concept makes it possible to combine such indices with other "cost" elements in a single metric (function) to be used for the evaluation of elicited roles during the role mining process. Minimizing such a function means eliciting those roles that contextually minimize the overall administration effort and fit the needs of an organization from a business perspective. This approach is particularly suitable for role mining algorithms since it makes it possible to introduce business elements within the analyzed data, thus leading to a hybrid role engineering approach. Note though that identifying cost elements and then combining them in a cost function is something that can be done in several ways. This, however, is a completely separate subject of research and, due to the page limit, it is only marginally addressed here (see Section 4.3). Instead, we mainly focus on the formal description of business elements that can contribute in defining a suitable cost function.

4.1 Activity-Spread

We now describe an index intended to evaluate business meaning. It is based on the analysis of business processes and their decomposition into activities. For this purpose, we observe that a typical organization is unlikely to have users that perform activities derived from different business processes. For example, given the processes "Human Resources Management" and "Inventory Management", it is very difficult that the organization needs a role that simultaneously allows activities within both these processes. This observation might also be applicable to the decomposition of a business process into simpler activities. It is difficult to have the same users involved in "Training and development" and "Recruitment" of employees, even though they are both activities of "Human Resources Management". However, this constraint becomes weaker as we compare simpler activities; for example, "Screening" and "Selection" might be two possible activities of "Recruitment", but it now becomes possible for a single user to perform both of them.

In general, given a role and the activities involved with it, the basic idea is that a role is likely to have a business meaning when such activities are "close" to one another within the process break-down structure. According to this, "Screening" and "Selection" are close since they are both children activities of "Recruitment"; instead, "Screening" is far from any other activity below "Inventory Management".

To measure the spreading of a role within the activity tree we resort to the farness concept introduced in Section 3.5. Given a role, farness may be used to evaluate the distances among activities granted by the role. After having calculated the farness index for each involved activity, averaging such indices offers a metric to capture the "degree of spread" of the role among its activities. We refer to such an index as *activity-spread* of a role. The higher the activity-spread is, the less business meaning the role has.

Distance Function. Before describing the activity-spread index in a formal way, we need to introduce the *distance* among two activities $a, a' \in ACTVT$ as:

$$d_{\text{actvt}}(a, a') = w_{\text{actvt}}(a \succeq \bar{a}) + w_{\text{actvt}}(a' \succeq \bar{a}),$$
$$\bar{a} = \{\alpha \in (\uparrow a \cap \uparrow a') \mid \forall \alpha' \in (\uparrow a \cap \uparrow a') : \alpha \succeq \alpha'\}, \quad (4)$$

whereas $\bar{a}$ is the nearest common parent of a, a'. If weights between activities are always equal to 1, the following alternative distance definition can be given:

$$d_{\text{actvt}}(a, a') = |\uparrow a| + |\uparrow a'| - 2\,|\uparrow a \cap \uparrow a'|. \quad (5)$$

Since activities are organized as a tree with a unique root, Equation 5 represents the number of edges between a and a'. Indeed, the distance between two vertices is the number of edges in the geodesic connecting them. Furthermore, it can be easily shown that the activity set is a metric space, since the provided function d_{actvt} is a metric; that is, given activities $a, a', a'' \in ACTVT$ the following properties holds:

- Distance is positive between two different activities, and precisely zero from an activity to itself, namely: $d_{\text{actvt}}(a, a') \geq 0$, and $d_{\text{actvt}}(a, a') = 0 \iff a = a'$.
- The distance between two activities is the same in either directions, namely: $d_{\text{actvt}}(a, a') = d_{\text{actvt}}(a', a)$.
- The distance between two activities is the shortest one along any path, namely: $d_{\text{actvt}}(a, a'') \leq d_{\text{actvt}}(a, a') + d_{\text{actvt}}(a', a'')$ (*triangle inequality*).

Activity-Spread Formalization. Given a role $r \in ROLES$, we identify the set of activities allowed by the role r as

$$\mathcal{A}_r = \{a \in ACTVT \mid \nexists a' \in (\downarrow a) \setminus \{a\},$$
$$authorized\_perms(r) \supseteq actvt\_perms^*(a),$$
$$authorized\_perms(r) \supseteq actvt\_perms^*(a')\}. \quad (6)$$

Equation 6 requires that *all* the permissions needed to execute the activities of $\mathcal{A}_r$ must be assigned to the role r. Thus, $\mathcal{A}_r$ contains only those activities that are allowed by assigning a user just to the role r. Further, given $a \in \mathcal{A}_r$ none of the parents of a are contained in $\mathcal{A}_r$, that is $\mathcal{A}_r$ contains only activities that are farther from the root.

We therefore define the *activity farness* for $a \in \mathcal{A}_r$ as:

$$d_{\mathcal{A}_r}(a) = \frac{1}{|\mathcal{A}_r| - 1} \sum_{a' \in \mathcal{A}_r} d_{\text{actvt}}(a, a'). \quad (7)$$

Equation 7 directly derives from Equation 3 (see Section 3.5) by adopting the distance function d_{actvt}. The greater $d_{\mathcal{A}_r}(a)$ is, the farther a is from all other activities allowed by permissions assigned to r.

Now we define a metric that takes into consideration the farness generated among *all* activities in $\mathcal{A}_r$. To this aim, the *variance* concept is likely to be the most intuitive and suitable to use. Given the arithmetic mean of all the farness indices calculated over $\mathcal{A}_r$, namely

$$\bar{d}_{\mathcal{A}_r} = \frac{1}{|\mathcal{A}_r|} \sum_{a \in \mathcal{A}_r} d_{\mathcal{A}_r}(a)$$
$$= \frac{1}{|\mathcal{A}_r|(|\mathcal{A}_r| - 1)} \sum_{a, a' \in \mathcal{A}_r} d_{\text{actvt}}(a, a'), \quad (8)$$

we define the *activity-spread* of a role $r \in ROLES$ as the farness variance among all the activities in $\mathcal{A}_r$, that is:

$$actvt\_spread(r) = \left(\frac{1}{|\mathcal{A}_r|}\sum_{a\in\mathcal{A}_r} d^2_{\mathcal{A}_r}(a)\right) - \bar{d}^2_{\mathcal{A}_r}. \quad (9)$$

Note that the value of $actvt\_spread(r)$ should be adjusted in order to assign a higher value to those roles associated with permissions that do not allow the execution of any activity. This way, it is possible to "dissuade" a role engineering process from eliciting roles where the corresponding permissions are not associated to any activities. Indeed, when a role contains permissions that are not related to any activities, it becomes harder to identify a business meaning for it. The number of permissions not related to activities are:

$$\left|authorized\_perms(r) \setminus \bigcup_{a\in\mathcal{A}_r} actvt\_perms(a)\right|. \quad (10)$$

If Equation 10 is equal to 0, there is no need to adjust $actvt\_spread(r)$. Otherwise, $actvt\_spread(r)$ may be multiplied by a coefficient that is proportional to Equation 10.

4.2 Organization-Unit-Spread

We now describe an index to evaluate business meaning of roles by analyzing the organization unit structure. With this aim in mind, note that users located into different OUs are likely to perform different tasks. Moreover, a role used across multiple organization units may require the co-ordination of various administrators; thus, requiring a higher administration effort than one for roles used exclusively within a single organizational unit. Hence, the idea is that the more distant the involved organization units are from each other, the less business meaning the role has. The ideal situation is when a role is almost exclusively assigned with users belonging to the "most central" OUs. Similar to the previous section, we calculate a farness index for each OU involved by a role—namely those OUs which contain the users assigned to this role. After having calculated the farness index for each involved OU, averaging such indices offers a metric to capture the "degree of spread" of the role among OUs. We refer to such an index as *organization-unit-spread* of a role. The higher the organization-unit-spread is, the less business meaning the role has.

Distance Function. Before formally describing the organization-unit-spread index, we need to introduce the *distance* among two organization units $o, o' \in OU$ as:

$$d_{\text{ou}}(o,o') = \text{w}_{\text{ou}}(o \succeq \bar{o}) + \text{w}_{\text{ou}}(o' \succeq \bar{o}),\ \bar{o} = \{\omega \in (\uparrow o \cap \uparrow o') \mid \forall \omega' \in (\uparrow o \cap \uparrow o') : \omega \succeq \omega'\}, \quad (11)$$

whereas $\bar{o}$ is the nearest common parent of o, o'. If weights between organization units are always equal to 1, the following alternative distance definition can be given:

$$d_{\text{ou}}(o,o') = |\uparrow o| + |\uparrow o'| - 2\,|\uparrow o \cap \uparrow o'|. \quad (12)$$

Since organizational units are organized as a tree with a unique root, Equation 12 represents the number of edges between o and o'. Furthermore, given $o, o', o'' \in OU$, for both distance definitions it can be demonstrated that:

- Distance is positive between two different organization units, and precisely zero from an organization unit to itself: $d_{\text{ou}}(o,o') \geq 0$, and $d_{\text{ou}}(o,o') = 0 \iff o = o'$.
- The distance between two organization units is the same in either direction: $d_{\text{ou}}(o,o') = d_{\text{ou}}(o',o)$.
- The distance between two organization units is the shortest one along any path: $d_{\text{ou}}(o,o'') \leq d_{\text{ou}}(o,o') + d_{\text{ou}}(o',o'')$ (*triangle inequality*).

Organization-Unit-Spread Formalization. Note that the number of users supporting each organization unit can influence the administration cost of roles. Indeed, bigger OUs require more effort from a single administrator or perhaps require multiple administrators. For this reason, the spreading index must be influenced both by the organization structure and by the percentage of users assigned to a given role and contextually belonging to the same organization unit.

Given a role $r \in ROLES$, we define the following sets:

- $\mathcal{O}_r = \{o \in OU \mid \nexists o' \in (\downarrow o) \setminus \{o\}, authorized\_users(r) \supseteq ou\_users^*(o), authorized\_users(r) \supseteq ou\_users^*(o')\}$, the set of organization units being involved with the role r.
- $\mathcal{U}_r = authorized\_users(r)$, the set of users assigned to r.
- Given $o \in \mathcal{O}_r$, then $\mathcal{U}_o = ou\_users^*(o)$ is the set of users assigned to an organization unit o.

Given $o \in \mathcal{O}_r$ none of the parents of o is contained in $\mathcal{O}_r$, that is $\mathcal{O}_r$ contains only OUs that are farther from the root. We therefore define the *OU farness* index for $o \in \mathcal{O}_r$ as:

$$d_{\mathcal{O}_r}(o) = \frac{1}{|\mathcal{U}_r|}\sum_{o'\in\mathcal{O}_r} |\mathcal{U}_{o'} \cap \mathcal{U}_r|\, d_{\text{ou}}(o,o'). \quad (13)$$

We assume that $(1/|\mathcal{U}_r|)\sum_{o\in\mathcal{O}_r} |\mathcal{U}_o \cap \mathcal{U}_r| = 1$, namely organization units in $\mathcal{O}_r$ contain all the users assigned to r and no user simultaneously belongs to more than one OU. The greater $d_{\mathcal{O}_r}(o)$ is, the farther o is from OUs containing the majority of users assigned to r. For example, if there is one particular OU containing most of the users of the role, then $d_{\mathcal{O}_r}(o)$ is close to the distance between o and such an OU.

Now we define the *variance* of all farness indices related to organization units in $\mathcal{O}_r$. Given the weighted arithmetic mean of all the farness indices calculated upon $\mathcal{O}_r$, namely

$$\bar{d}_{\mathcal{O}_r} = \frac{1}{|\mathcal{U}_r|}\sum_{o\in\mathcal{O}_r} |\mathcal{U}_o \cap \mathcal{U}_r|\, d_{\mathcal{O}_r}(o) = \frac{1}{|\mathcal{U}_r|^2}\sum_{o,o'\in\mathcal{O}_r} |\mathcal{U}_o \cap \mathcal{U}_r|\,|\mathcal{U}_{o'} \cap \mathcal{U}_r|\, d_{\text{ou}}(o,o'), \quad (14)$$

the *organization-unit-spread* is defined as

$$ou\_spread(r) = \left(\frac{1}{|\mathcal{U}_r|}\sum_{o\in\mathcal{O}_r} |\mathcal{U}_o \cap \mathcal{U}_r|\, d^2_{\mathcal{O}_r}(o)\right) - \bar{d}^2_{\mathcal{O}_r}. \quad (15)$$

Note that the value of $ou\_spread(r)$ grows when: OUs contain many users—identified through $|\mathcal{U}_o \cap \mathcal{U}_r|$; or, OUs are far from each other—according to $d^2_{\mathcal{O}_r}(o)$.

4.3 Cost Function

In this section we briefly explain how cost elements can be combined in a global cost function, thus leveraging our cost-driven approach (see Section 3.2). In general, finding the optimal candidate role-set can be seen as a *multi-objective optimization problem* [8]. An optimization problem is multi-objective when there are a number of objective functions that are to be minimized or maximized. Multi-objective optimization often means to trade-off conflicting goals. In a role engineering context, possible objectives are:

- Minimize the number of roles;
- Minimize the number of role possessed by each user;
- Maximize the business meaning of roles, that corresponds to minimizing the activity-spread and/or the organization-unit-spread indices of all elicited roles.

The previous statements can be formalized as:

- $\min\{|ROLES|\}$;
- $\min\left\{\sum_{r\in ROLES} assigned\_users(r)\right\} = \min\{|UA|\}$;
- $\min\left\{\sum_{r\in ROLES} actvt\_spread(r)\right\}$;
- $\min\left\{\sum_{r\in ROLES} ou\_spread(r)\right\}$.

The constraint of this optimization problem is that elicited roles must "cover" all possible combinations of permissions, namely they represent a candidate role-set.

The dependencies among these objectives are quite complex. For example, if on one side $|ROLES|$ decreases, there is a strong chance that more roles will be required to cover all the permissions possessed by users, causing $|UA|$ to increase. Since each role has a high number of assigned users, it is likely that the number of involved organization units is high, thus increasing the value of the *ou_spread* function. On the other hand, if we want to reduce the average number of roles per user, we will need more *ad-personam* roles, then $|ROLES|$ will likely increases.

Hence, there is little use in the quest for a global maximum or a global minimum. Since trade-off for conflicting criteria can be defined in many ways, there exist multiple approaches to define what an optimum is. The simplest approach is that of computing a *weighted sum* [8]: multiple objectives are transformed into an aggregated scalar objective function by multiplying each objective by a weighted factor and summing up all contributors. As for role engineering, we can combine all the proposed objectives as follows:

$$\min\Big\{w_r|ROLES| + w_u|UA| + w_a \sum_{r\in ROLES} actvt\_spread(r) + w_o \sum_{r\in ROLES} ou\_spread(r)\Big\}. \quad (16)$$

where the w_i indicate the importance of the i^{th} objective in the overall optimization problem. Section 5.2 will show a real application of Equation 16 to a role mining algorithm.

5. DISCUSSION AND EXAMPLES

In this section we will analyze the distinguishing features of the proposed framework through some practical examples. We first show a use case of the activity-spread index via a simple example. Then, we demonstrate the effectiveness of the organization-unit-spread index through the RBAM role mining algorithm [2] applied to an existing company. Due to space limitation, we do not provide examples of the combined usage of both indices.

5.1 Example of Activity-Spread

Figure 2 depicts a possible candidate role set and an activity tree. At the top there are representations of RBAC entities, namely roles and permissions. Example roles are $assigned\_perms(r_1) = \{p_1, p_2\}$, $assigned\_perms(r_6) = \{p_3\}$, while $authorized\_perms(r_6) = \{p_1, p_2, p_3, p_4\}$ since $r_1 \succeq r_6$ and $r_2 \succeq r_6$. At the bottom of the figure, there are activities organized in a tree structure. For instance, a_{11} is the root of

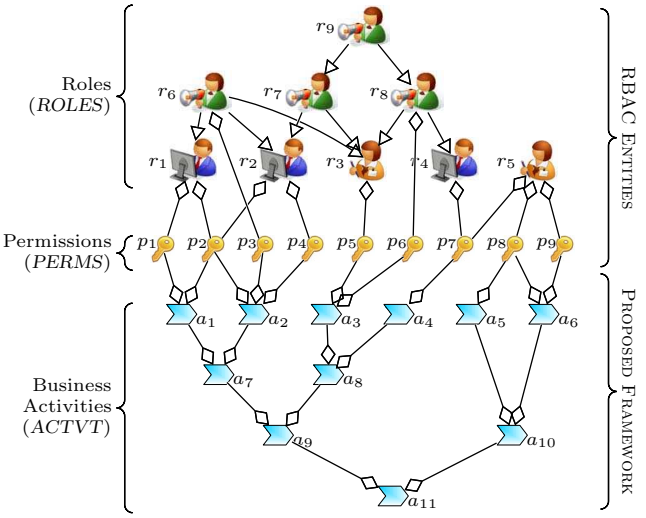

Figure 2: An example of activity model

the tree, while $a_9 \rightarrow a_{11}$ and $a_7 \succeq a_{11}$ holds. To ease exposition, weights of connections between activities are not represented in Figure 2, and they are all assumed to be equal to 1. Further, $\downarrow a_8 = \{a_3, a_4, a_8\}$ and $\uparrow a_1 = \{a_1, a_7, a_9, a_{11}\}$. Activity a_7 is performed when a user possesses the permissions p_1, p_2 (that allows for the child activity a_1) and p_2, p_3, p_4 (that are related to the child activity a_2).

As for the business meaning estimation, let us calculate the activity-spread of role r_6. By looking at Figure 2 it is easy to verify that role r_6 allows for the execution of the activities $\mathcal{A}_{r_6} = \{a_1, a_2\}$. In line with Equation 7, the farness index for each activity in $\mathcal{A}_{r_6}$ is:

$$d_{\mathcal{A}_{r_6}}(a_1) = \frac{d_{\text{actvt}}(a_1,a_1)+d_{\text{actvt}}(a_1,a_2)}{2-1} = \frac{0+2}{2-1} = 2,$$
$$d_{\mathcal{A}_{r_6}}(a_2) = \frac{d_{\text{actvt}}(a_2,a_1)+d_{\text{actvt}}(a_2,a_2)}{2-1} = \frac{2+0}{2-1} = 2.$$

For such indices we have $\bar{d}_{\mathcal{A}_{r_6}} = \frac{2+2}{2} = 2$ and thus:

$$actvt\_spread(r_6) = \frac{2^2+2^2}{2} - 2^2 = 0.$$

The activity spread is 0 since the activities associated to r_6 are somewhat "balanced" within the tree. Thus, r_6 is probably a good role; indeed, it allows the execution of activities a_1, a_2, and consequently the execution of a_7.

A different result is obtained by analyzing role r_5. In this case, $\mathcal{A}_{r_5} = \{a_4, a_5, a_6\}$. Then:

$$d_{\mathcal{A}_{r_5}}(a_4) = \frac{d_{\text{actvt}}(a_4,a_4)+d_{\text{actvt}}(a_4,a_5)+d_{\text{actvt}}(a_4,a_6)}{4-1} = \frac{0+5+5}{3-1} = 5,$$
$$d_{\mathcal{A}_{r_5}}(a_5) = \frac{d_{\text{actvt}}(a_5,a_4)+d_{\text{actvt}}(a_5,a_5)+d_{\text{actvt}}(a_5,a_6)}{4-1} = \frac{5+0+2}{3-1} = \frac{7}{2},$$
$$d_{\mathcal{A}_{r_5}}(a_6) = \frac{d_{\text{actvt}}(a_6,a_4)+d_{\text{actvt}}(a_6,a_5)+d_{\text{actvt}}(a_6,a_6)}{4-1} = \frac{5+2+0}{3-1} = \frac{7}{2}.$$

Notice that a_4 has the largest farness, hence it is far from other activities. These observations can also be graphically justified by observing Figure 1. The arithmetic mean of such indices is $\bar{d}_{\mathcal{A}_{r_5}} = \frac{5+7/2+7/2}{3} = 4$ and thus:

$$actvt\_spread(r_5) = \frac{(5)^2+(7/2)^2+(7/2)^2}{3} - 4^2 = 0.5.$$

This means that r_5 has less business meaning than r_6, since r_5 is more spread out among its activities.

Finally, it is worth noticing that both the farness index and the activity-spread make up useful information that

Organization Unit Tree	Users			
	$w_o = 0$	$w_o = 1$		
	Role107	Role107	Role752	Role293
ORGANIZATION				
EMPLOYEES				
CORPORATE				
ADMINISTRATION PLANNING & CONTROL				
FISCAL ADMINISTRATION	2		2	
PLANNING CONTROL SERVICES & STAFF				
PC/SERVICE AREA & STAFF-ICT				
PC/SERVICE AREA	2			2
INTERCOMPANY SERVICES AND OTHER ACTIVITIES				
SERVICES AND REAL ESTATE				
NORTH WEST AREA				
PLANNING AND GENERAL SERVICES	1	1		
TOTAL USERS	5	1	2	2

Figure 3: An example of elicited roles with different values for the weight w_o

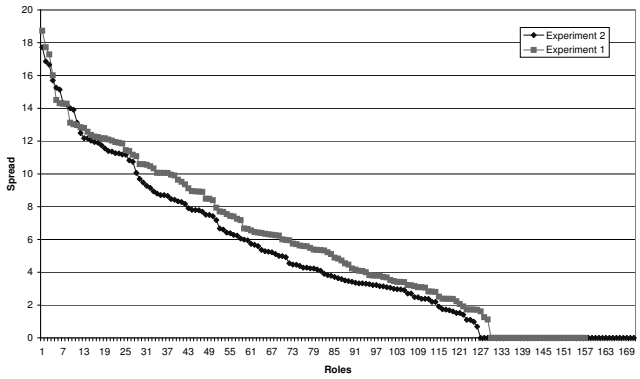

Figure 4: Roles obtained via different values of w_o and orderer by descending organization-unit-spread

could also be visualized in a role engineering tool for each activity that is involved within a given role. Having access to this information could help users validate roles from a business perspective.

5.2 Organization-Unit-Spread on Real Data

To highlight the framework viability in real applications, we examined a large private company. In particular, we analyzed permissions related to an ERP application. The system configuration was made up of 1,139 permissions that were used by 1,034 users within 231 organization units, resulting in 10,975 user-permission assignments.

To test the effectiveness of our approach, we used an improved version of the RBAM algorithm (see [2,3] for further details), seeking to elicit roles which minimize the function

$$c = w_r|ROLES| + w_u|UA| + w_o \sum_{r \in ROLES} ou\_spread(r). \quad (17)$$

The algorithm was set to discard permission combinations possessed by less than 4 users. Moreover, we assigned a weight 1 to all direct hierarchical relationships between organization units. Then, we compared the algorithm output obtained in two distinct settings: in the first one (from now on indicated as "Experiment 1"), we did not considered the contribution of the organization-unit-spread index, by using $w_o = 0$ and $w_r = 10$, $w_u = 1$; in the second one (from now on indicated as "Experiment 2"), we considered all the cost elements provided by Equation 17, using $w_o = 1$ and the same values for the other weights, namely $w_r = 10$, $w_u = 1$.

The first thing that we can observe from Experiment 2 is that the number of elicited roles is greater than the number of elicited roles in Experiment 1. In particular, we have 157 roles in the first case (that cover 7,857 user-permission assignments with 2,191 role-user relationships) and 171 roles in the second (that cover 7,857 user-permission assignments with 2,196 role-user relationships). The justification for this behavior is that when using $w_o \neq 0$ (namely, taking into account the organization-unit-spread) in some cases it is necessary to reduce the number of users assigned to a role by introducing additional roles to be assigned with a subset of such users. Indeed, the greater the number of users assigned to a role are, the more organization units are likely to be involved. Hence, according to Equation 15, the value of the function *ou_spread* is likely to be higher.

Figure 3 shows an interesting example—to protect company privacy, the names of the organization units are slightly different from the original ones. In Experiment 1 ($w_o = 0$), we noticed that 2 out of 1,139 permissions were possessed by 22 users, and these users were granted these permissions by assigning them to only one of the two roles which the algorithm automatically called `role107` and `role594`. Moreover, $\texttt{role594} \succeq \texttt{role107}$, while $|authorized\_perms(\texttt{role107})| = 2$ and $|authorized\_perms(\texttt{role594})| = 15$. Further, we have that $|assigned\_users(\texttt{role107})| = 5$ and users were belonging to 3 different organization units, namely 'FISCAL ADMINISTRATION', 'PC/SERVICE AREA' and 'PLANNING AND GENERAL SERVICES'. Because of the spreading of users among these OUs, $ou\_spread(\texttt{role107}) = 12.243$. Figure 3 also offers a partial view of the entire organization structure, showing the already cited organization units with a bold font. Conversely, in Experiment 2 (where $w_o = 1$) the same 2 permissions were completely covered by 4 roles that the algorithm automatically called `role107`, `role752`, `role293`, `role594`. Roles with the same name between the two experiments contain the same permission set, but are assigned with different user sets. Indeed, by observing Figure 3 it is possible to verify that the same 5 users from the other experiment now spread among 3 roles, having $|assigned\_users(\texttt{role107})| = 1$, $|assigned\_users(\texttt{role752})| = 2$, and $|assigned\_users(\texttt{role293})| = 2$. In this case, each of these roles have assigned users who belong to only one organization unit. Moreover, $\texttt{role752} \succeq \texttt{role107}$ and $\texttt{role293} \succeq \texttt{role107}$. One role has $|authorized\_perms(\texttt{role752})| = 13$ and the other one has $|authorized\_perms(\texttt{role293})| = 8$. Their organization-unit-spread is equal to 0, thus demonstrating that taking the organization-unit-spread into account may help identify roles with higher business meaning.

Another interesting behavior of the RBAM algorithm, if used with the cost function described in Equation 17, is that increasing the value of w_o causes the decrease of the average organization-unit-spread. Indeed, when $w_o = 0$ we have an average value of $\left(\sum_{r \in ROLES} ou\_spread(r)\right)/|ROLES| = 5.96$. Instead, when $w_o = 1$ the average spread drops down to 4.86, and $w_o = 10$ causes an average spread of 2.87. This means that a high value of w_o allows to elicit roles with a more relevant business meaning on average. In such a case, the price to pay to obtain more meaningful roles is in terms of the cardinality of *ROLES*. By analyzing Figure 4, it could also be noted that even though Experiment 2 has more roles,

the percentage of roles with spread equal to 0 is increased in comparison with Experiment 1.

6. CONCLUDING REMARKS

This paper provides a new formal framework for the role engineering problem. It allows to implement a concrete hybrid approach through the combination of existing role mining algorithms (bottom-up) and business analysis (top-down). Once the business processes of an organization have been correctly modeled, we then analyze roles in order to evaluate their business meaning. In particular, a role is likely to be meaningful when it involves activities within the same business process. Thus, we measure the spreading of a role among business processes by introducing the spread index. Leveraging the cost-driven approach makes it possible to take into account the spread during the role engineering process. The proposed framework has been implemented on a real case, and the results support the its viability.

As for future work, we are currently striving to identify other business data that can be used to estimate the business meaning of elicited roles.

7. REFERENCES

[1] American National Standards Institute (ANSI) and InterNational Committee for Information Technology Standards (INCITS). ANSI/INCITS 359-2004, Information Technology – Role Based Access Control, 2004.

[2] A. Colantonio, R. Di Pietro, and A. Ocello. A cost-driven approach to role engineering. In *Proceedings of the 23rd ACM Symposium on Applied Computing, SAC '08*, volume 3, pages 2129–2136, Fortaleza, Ceará, Brazil, 2008.

[3] A. Colantonio, R. Di Pietro, and A. Ocello. Leveraging lattices to improve role mining. In *Proceedings of the IFIP TC 11 23rd International Information Security Conference, SEC '08*, volume 278 of *IFIP International Federation for Information Processing*, pages 333–347. Springer, 2008.

[4] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. Verde. Mining stable roles in RBAC. In *Proceedings of the IFIP TC 11 24th International Information Security Conference, SEC '09*, volume 297 of *IFIP International Federation for Information Processing*, pages 259–269. Springer, 2009.

[5] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. Verde. A probabilistic bound on the basic role mining problem and its applications. In *Proceedings of the IFIP TC 11 24th International Information Security Conference, SEC '09*, volume 297 of *IFIP International Federation for Information Processing*, pages 376–386. Springer, 2009.

[6] E. J. Coyne. Role-engineering. In *Proceedings of the 1st ACM Workshop on Role-Based Access Control, RBAC '95*, pages 15–16, 1995.

[7] R. Crook, D. Ince, and B. Nuseibeh. Towards an analytical role modelling framework for security requirements. In *Proceedings of the 8th International Workshop on Requirements Engineering: Foundation for Software Quality, REFSQ '02*, 2002.

[8] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[9] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan. Fast exact and heuristic methods for role minimization problems. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, SACMAT '08*, pages 1–10, 2008.

[10] P. Epstein and R. S. Sandhu. Engineering of role/permission assignments. In *Proceedings of the 17th Annual Computer Security Applications Conference, ACSAC*, pages 127–136. IEEE Computer Society, 2001.

[11] E. B. Fernandez and J. C. Hawkins. Determining role rights from use cases. In *Proceedings of the 2nd Workshop on Role-Based Access Control, RBAC '97*, pages 121–125, 1997.

[12] M. Frank, D. Basin, and J. M. Buhmann. A class of probabilistic models for role engineering. In *Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS '08*, Oct. 2008.

[13] 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 the 7th ACM Symposium on Access Control Models and Technologies, SACMAT '02*, 2002.

[14] M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining – revealing business roles for security administration using data mining technology. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies, SACMAT '03*, pages 179–186, 2003.

[15] N. Li, J.-W. Byun, and E. Bertino. A critique of the ANSI standard on role-based access control. *IEEE Security & Privacy*, 5(6):41–49, 2007.

[16] H. Lu, J. Vaidya, and V. Atluri. Optimal boolean matrix decomposition: Application to role engineering. In *Proceedings of the 24th IEEE International Conferene on Data Engineering, ICDE '08*, pages 297–306, 2008.

[17] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo. Mining roles with semantic meanings. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, SACMAT '08*, pages 21–30, 2008.

[18] G. Neumann and M. Strembeck. A scenario-driven role engineering process for functional RBAC roles. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, SACMAT '02*, pages 33–42, 2002.

[19] S. Oh and S. Park. Task-role-based access control model. *Information Systems*, 28(6):533–562, 2003.

[20] S. Oh, R. S. Sandhu, and X. Zhang. An effective role administration model using organization structure. *ACM Transactions on Information and System Security, TISSEC*, 9(2):113–137, 2006.

[21] N. Perwaiz and I. Sommerville. Structured management of role-permission relationships. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies, SACMAT '01*, pages 163–169, 2001.

[22] H. Röckle, 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 the 5th ACM Workshop on Role-Based Access Control, RBAC 2000*, pages 103–110, 2000.

[23] J. Schlegelmilch and U. Steffens. Role mining with ORCA. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies, SACMAT '05*, pages 168–176, 2005.

[24] D. Shin, G.-J. Ahn, S. Cho, and S. Jin. On modeling system-centric information for role engineering. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies, SACMAT '03*, pages 169–178, 2003.

[25] J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, SACMAT '07*, pages 175–184, 2007.

[26] J. Vaidya, V. Atluri, and J. Warner. RoleMiner: mining roles using subset enumeration. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 144–153, 2006.

[27] S. Wasserman and K. Faust. *Social Network Analysis*, chapter 5, pages 169–219. Cambridge University Press, 1994.

[28] D. Zhang, K. Ramamohanarao, and T. Ebringer. Role engineering using graph optimisation. In *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, SACMAT '07*, pages 139–144, 2007.