ORGODEX

A New Paradigm for Role-Based Access Control

ORGODEX

Un nouveau paradigme pour le contrôle d'accès basé sur les rôles

A Thesis Submitted to the Division of Graduate Studies
of the Royal Military College of Canada
by

Aaron A. Elliott, MSc

In Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

May 2018

Acknowledgements

I thank God for the blessings bestowed upon me. I thank my family for their love, support and encouragement.

Linda, Nicholas and Alexander, you are everything to me. Together, we work hard to find balance each and every day, sharing our successes, loving and supporting one another unconditionally. More than anything I am proud of you and what we accomplish as a family.

Mom and Dad you are my foundation and I love you dearly. I strive to make you proud every day. You are a shining example, inspiring me to be better.

Scott and Tamara I treasure our time together. My brother, my sister, my best friends, time spent with you and your families are amongst my fondest memories. I look forward to future fun and adventures, smiling as I write this.

I would like to express my sincere gratitude to my supervisor, Scott Knight. Without your guidance and direction I calculate my probability of successfully completing this dissertation in the range of nil to improbable. I have the utmost respect for you and I thank Terry Shepard for connecting us many years ago.

I would like to thank my colleagues at the Royal Military College of Canada who continually challenge me with their honest uncensored opinions. You have heavily influenced my work. For the friendships forged and the many insightful conversations, I am eternally grateful.

I would like to thank the members of my examining committee for their valuable review and feedback, further strengthening this dissertation.

Abstract

Role-Based Access Control (RBAC) is a popular solution for implementing information security however there is no pervasive methodology used to produce scalable access control systems for large organizations with hundreds or thousands of employees. As a result, ten engineers will likely arrive at ten different solutions to the same problem where there is no right or wrong answer but the cost is both immediate and long term. Moreover, they would have difficulty communicating the important aspects of their design implementations to each other. This is an interesting deficiency because despite their diversity, large organizations share two key concepts, roles and responsibilities, where a role like Departmental Chair has responsibilities. In this work, our objective is to introduce ORGODEX, a new model and practical methodology for engineering scalable RBAC systems in large organizations where employees require access to information on a need to know basis. First, we motivate the requirement for new structural RBAC relationships, distinguishing between roles and responsibilities. Next, we introduce our new model for describing and reasoning about RBAC implementations. Then we produce a new iterative methodology for engineering scalable access control systems. Finally, we validate our work with a case study whereby the ORGODEX model and methodology are used to deploy authorization as a service in the context of cloud computing.

Résumé

Le contrôle d'accès basé sur les rôles (CABR) est une solution populaire pour la mise en œuvre de la sécurité de l'information, mais il n'existe pas de méthodologie omnipreésente utilisée pour produire des systèmes de contrôle d'accès évolutifs pour les grandes organisations avec des centaines ou des milliers d'employés. En conséquence, dix ingénieurs arriveront probablement à dix solutions différentes pour le même problème où il n'y a pas de bonne ou mauvaise réponse, mais où il existe à la fois un coût immédiat et à long terme. En outre, ils auront du mal à communiquer les aspects importants de la conception de leurs implémentations. Il s'agit d'une lacune intéressante car, malgré leur diversité, les grandes organisations sont fondées sur deux concepts clés, des rôles et des responsabilités, où un rôle comme chef de département est identifié et attribué des responsabilités. Dans ce travail, notre objectif est d'introduire ORGODEX, un nouveau modèle et une méthodologie concrète pour l'ingénierie des systèmes évolutifs CABR dans les grandes organisations où les employés ont besoin d'accès à de l'information sur le principe du besoin de savoir. Premièrement, nous motiverons l'exigence d'une nouvelle relation structurelle CABR, en distinguant les rôles et les responsabilités. Ensuite, nous présentons notre nouveau modèle pour décrire et raisonner les implémentations CABR. Ensuite, nous produisons une nouvelle méthodologie itérative pour l'ingénierie des systèmes de contrôle d'accès évolutifs. Enfin, nous validons notre travail avec une étude de cas selon laquelle le modèle et la méthodologie ORGODEX sont utilisés pour déployer l'autorisation en tant que service dans le contexte du cloud computing.

Contents

List of Tables

List of Figures

List of Abbreviations

AaaS	Authorization as a Service	100
ABAC	Attribute-Based Access Control	vi
AC3	Access Control for Cloud Computing	101
A-ERBAC	Administrative Enterprise Role-based Access Control	vi
AP	Administrative Permissions	17
AR	Administrative Roles	17
ARA	Ability Role Assignment	21
ARBAC	Administrative Role-Based Access Control	v
ARBAC97	Administrative Role-Based Access Control 1997	16
ARBAC99	Administrative Role-Based Access Control 1999	16
ARBAC02	Administrative Role-Based Access Control 2002	16
AR-UG	Associate Registrar, Undergraduate	62
AR-PG	Associate Registrar, Postgraduate	62
BA	Business Analyst	75
C	Constraints	14
CEO	Chief Executive Officer	46
DC	Departmental Chair	54
DEV	Developer	101
DSD	Dynamic Separation of Duty	126
ER	Enterprise Roles	27
ERBAC	Enterprise Role-based Access Control	35
GoC	Government of Canada	1
GoC-SP	Government of Canada Security Policy	40
GRA	Group Role Assignment	21
HR	Human Resources	25
HRMS	Human Resources Management System	22
IM	Information Management	v
IT	Information Technology	7
MITS	Operational Security Standard: Management of Information Technology Security	1

MTA	Multi-Tenant Architecture	101
NIST	National Institute of Standards and Technology	12
OAuth	Open Authorization	70
OATT	Object Attributes	32
P	Permissions	2
PA	Permission Assignment	12
PD	Project Director	101
PDCA	Plan-Do-Check-Act	69
PFP	Permission Filtering Policy	32
PM	Project Manager	101
PR	Permission-Role	4
PRA	Permission-Role Assignment	vi
PRMS	Permissions	32
R	Roles	12
RABAC	Role-Centric Attribute Based Access Control	vi
RBAC	Role-Based Access Control	iii
RBAC96	Role-Based Access Control 1996	v
RBÄC	Bi-Sorted Role-Based Access Control	vi
RDBMS	Relational Database Management Systems	64
RE	Requirements Engineering	76
ReBAC	Relationship-Based Access Control	vi
ReMMo	Responsibility MetaModel	vi
RH	Role Hierarchy	4
RHA	Role Hierarchy Administration	25
RMP	Role Mining Problem	56
RRA	Role-Role Assignment	vi
S	Subjects	2
SA	Security Architect	81
SaaS	Software as a Service	100
SARBAC	Scoped Administrative Role-based Access Control	vi
SNS	Social Network Systems	30
SOR	Statement of Requirements	75
SP	Subject-Permission	2
SQL	Structured Query Language	95
SR	Subject-Role	4
SRP	Subject-Role-Permission	4
SRS	Software Requirements Specifications	77
SSD	Static Separation of Duty	126
TS	Target System	27
U	Users	12

UA	User Assignment	12
UATT	User Attributes	32
UIS	University Information System	70
UISA	University Information System Application	101
URA	User-Role Assignment	v
XACML	eXtensible Access Control Markup Language	69

1 Introduction

The research described in this dissertation proposes a new model and methodology for engineering scalable access control systems in organizations where hundreds or thousands of employees require access to information on a need to know basis in order to perform their job. In this chapter, we introduce the research area, identify an area of deficiency, state our aim and describe our validation approach.

1.1 Information Management (IM)

The Government of Canada (GoC) Policy on Information Management states that *the objective is to achieve efficient and effective information management to support program and service delivery; foster informed decision making; facilitate accountability, transparency, and collaboration; and preserve and ensure access to information and records for the benefit of present and future generations* [61].

Applications are developed to permit access to corporate information in dynamic, complex environments with challenging combinations of employee turnover, security policy and regulatory compliance. Figure 1.1 is one example of an IM architecture where several applications may access the corporate database directly, or indirectly through web or application servers. After a user successfully enters their user name and password, authenticating their identity, an application may limit their access to information, employing a *need to know* security policy.

Under section 16.4.3 of the GoC Operational Security Standard: Management of Information Technology Security (MITS), administrators must keep user access to the minimum required for an individual to perform their duties. Furthermore, administrators must ensure that access control implementations are regularly updated to accurately reflect the current responsibilities of any individual in the organization [59].

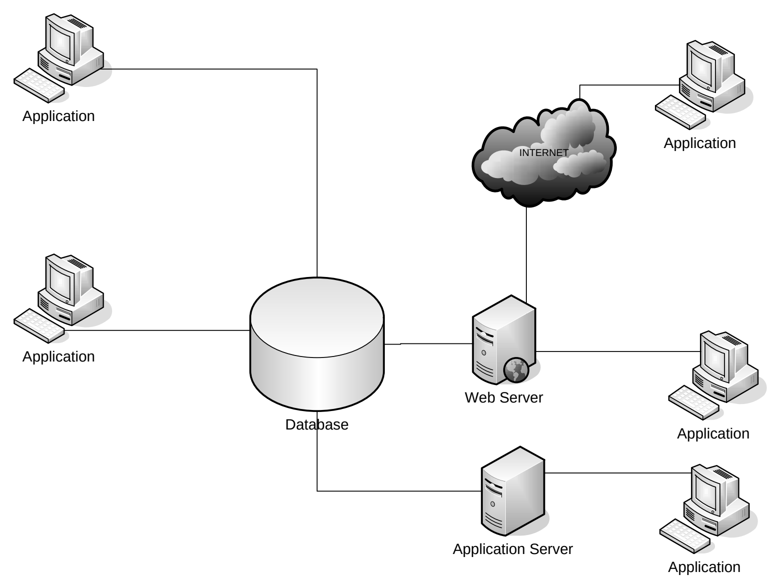

Figure 1.1: IM architectures may include several applications.

In this work the terms IM system and application are used interchangeably as are the terms subject and user.

1.2 Authorization

Access to information is controlled by layers of security. The authentication layer facilitates entry into the system, enabling subjects to use their key, for instance a user name and password, to open up the proverbial door. Once inside the information system, the authorization layer, specifies WHO has access to WHAT. The who is a set of Subjects (S) and the what is a set of Permissions (P) that have been assigned to the subject. The result is a set of Subject-Permission (SP) relationships, where the set SP $\subseteq$ S $\times$ P specifies who is authorized to do what in the information system (Figure 1.2). The authorization layer explicitly details all aspects of a security implementation for authenticated subjects, forming a rich, complex arena for security practi-

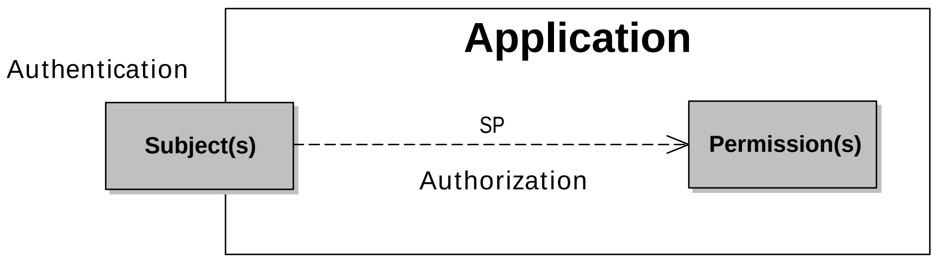

Figure 1.2: SP is the implicit relationship between Subjects (S) and Permissions (P).

tioners to manage.

Using an analogous scenario from a physical security implementation, consider the case of Bob an employee at a university, whose identification card acts like a key. Although Bob's key permits him to access particular buildings and rooms, Bob does not have full access to all buildings and rooms on campus. Security policy dictates that Bob only needs access to a subset of the buildings and rooms. Bob has no need to access student residences, nor many other buildings throughout the campus.

In this scenario, Bob's identification card is the authentication layer. Bob uses his card as a key, identifying himself to the access control system. Once his key has been presented, the authorization layer, determines whether Bob (WHO) may enter the building (WHAT). Maintaining WHO may do WHAT is not trivial in physical security implementations.

Engineering scalable access control systems permitting employees to utilize multiple applications with potentially hundreds or thousands of features and functionalities on a need to know basis is extremely challenging. In this work we assume a subject has successfully authenticated to an application, for instance by correctly entering their user name and password. After the subject has gained entry to the IM system, they are only permitted to read or write information as authorized. The focus of this dissertation is the authorization layer, where relations between WHO and WHAT are explicitly defined and maintained.

1.3 Role-Based Access Control (RBAC)

Since the early 1990's, the use of RBAC has evolved within large organizations such as the GoC. Although, RBAC provides a solid foundation for managing access control, system administrators may be burdened with the maintenance of hundreds or thousands of roles across several applications [76]. Managing these roles, users and their interrelationships is a formidable task that is often highly centralized in small teams of system administrators [74]. This is a daunting task in large organizations where security practitioners are tasked with access control as a secondary duty and provided various levels of training and formalized knowledge [9].

RBAC is a popular model for implementing the authorization layer or WHO may do WHAT [72]. The use of RBAC for access control has experienced a global proliferation due to the administrative savings afforded by the RBAC model [12][26][72]. Unlike the concept of groups, the administration of RBAC is multi-faceted. Performing Subject-Role (SR) enrollments, creating a Role Hierarchy (RH) and instantiating Permission-Role (PR) assignments are distinct administrative actions that explicitly relate S and P. An explicit representation includes at least one role between subjects and permissions, forming the classic Subject-Role-Permission (SRP) triangle (Figure 1.3).

Consider the case where all bank tellers require the exact same access to an information system, conservatively five permissions. In this classic RBAC example, a security officer, responsible for access management, creates the role *bank teller* (1), assigns five (5) permissions to the role *bank teller* and then enrolls a subject into the role *bank teller* (1). By creating the role and assigning it, the security officer invests two extra administrative actions compared to simply assigning five permissions directly to the subject, however, the return on investment is four fewer administrative actions for each additional subject enrolled into the *bank teller* role. In Figure 1.4, we see that the savings is fifteen administrative actions (less two), or grants, in case #1 with five subjects, or users. In large organizations like case #2, we see that the savings is 9,800 (less two) administrative actions for one hundred subjects.

With RBAC the administrative savings are easily demonstrable. Unfortunately, this example oversimplifies RBAC, making it difficult to understand the challenges inherent when implementing RBAC in large dynamic organizations with highly diversified workforces. For large-scale RBAC systems, thousands of access control relationships must be maintained and the administrative effort to maintain SR, RH and PR relationships is a formidable task that is often highly centralized in a small team of security administrators [63].

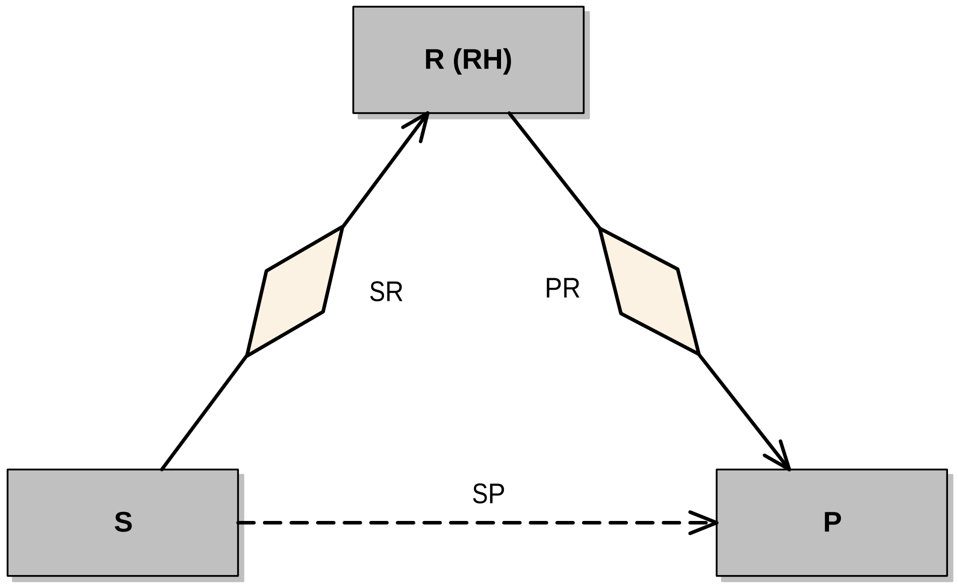

Figure 1.3: An explicit RBAC relationship includes at least one role, forming the classic subject-role-permission triangle.

1.4 Operational Case Studies

Protecting information and client data is an ongoing concern for large public and private organizations. However, the administration of large-scale RBAC systems like Dresdner Bank [76], with 40,000 users and 1300 roles, remains a challenging problem [51]. RBAC is an open, generic technology that permits several solutions to the same access control requirements and for this reason, large banking organizations like Dresdner may have similar access control requirements and vastly different RBAC systems.

Coyne suggests that the definition of roles is essentially a requirements engineering process [13]. This is an important concern since role engineering has been determined to be the most expensive aspect of deploying RBAC [58]. This research is particularly interested in the administration of RBAC systems in large organizations with hundreds or thousands of employees. This includes the design, implementation, deployment and maintenance of RBAC systems for information security. The relevance of this research is best illustrated with two operational case studies.

In the first case study, conducted from the year 2000 onwards, we analyze

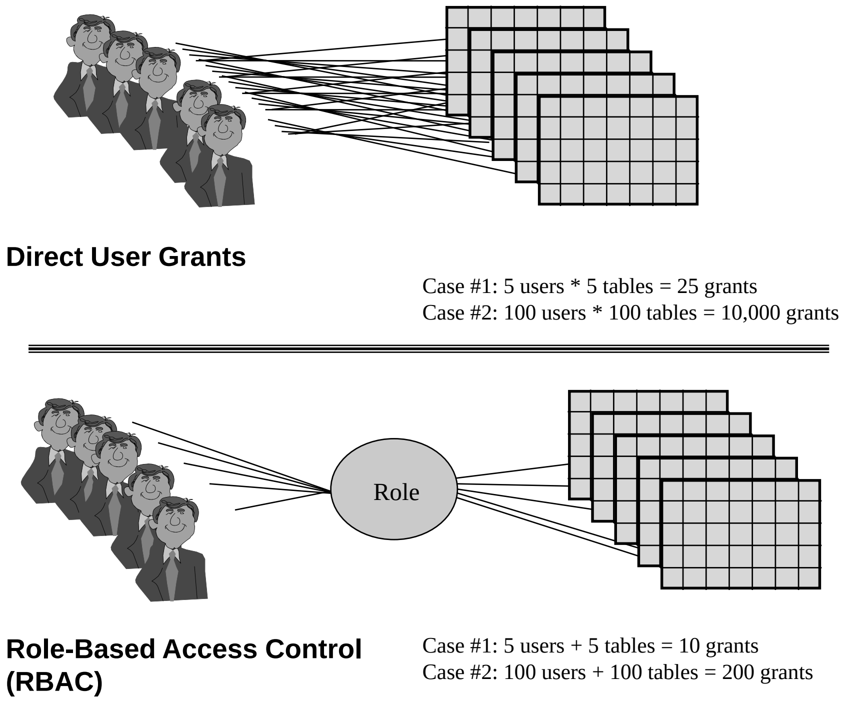
Direct User Grants
Case #1: 5 users * 5 tables = 25 grants
Case #2: 100 users * 100 tables = 10,000 grants
Role
Role-Based Access Control (RBAC)
Case #1: 5 users + 5 tables = 10 grants
Case #2: 100 users + 100 tables = 200 grants

Figure 1.4: Classic Example of RBAC Administrative Savings

the RBAC implementation of ACME university[1], resulting in a deep-seated appreciation of the inherent challenges faced when administering large RBAC implementations on a yearly basis. We have closely observed the impact of employee turnover and security policy changes in a real-world setting, forming the motivation for this research and influencing our work.

In the second case study, we describe a project conducted for MECA[2], an educational institution, whose campus is located more than three hundred kilometers from ACME university. The scope is a one year engagement (2016) to extend features and functionalities of the ACME university information system to MECA. We have closely observed the communication challenges encountered by a distributed, multi-disciplinary, multi-lingual project team with extensive to very little understanding of access control.

[1]The name ACME University is inspired from ACME Looniversity http://tinytoons.wikia.com/wiki/Acme_Looniversity

[2]The name MECA is an anagram of ACME

1.5 Statement of Deficiency

RBAC is a popular solution for implementing access control however our research suggests there is no pervasive methodology used to produce scalable access control systems for large organizations with hundreds or thousands of employees. As a result ten engineers will likely arrive at ten different solutions to the same problem where there is no right or wrong answer but there is both an immediate and long term cost. Moreover, they would have difficulty communicating the important aspects of their design and implementations to one another. This is an interesting deficiency because despite their diversity, large organizations are built upon two key concepts, roles and responsibilities, where a role like Departmental Chair is identified and assigned responsibilities.

Access control is often considered a necessary burden. Organizations must secure their data in order to preserve intellectual property or meet privacy concerns and too often RBAC systems are considered a means of restricting access to information. This is disappointing because RBAC has the potential to be a well understood, enabling technology, where implementations directly reflect the business model of an organization. Instead RBAC systems are often loosely coupled with the business model, ripe with redundancies and costly to maintain.

This is an important concern because role engineering is expensive [76][58][1]. Large organizations who have significant investments in RBAC implementations with hundreds of roles engineered over decades to realize security policy and safeguard information assets would like to reuse their implementation when deploying new services and adapting to meet the demands and expectations of clients, stakeholders, partners and employees. For large, mature organizations with decades of Information Technology (IT) investments, the challenge of replacing legacy solutions with modern alternatives are well-documented and costly [37][70][77]. IT supports IM which must support the business [82].

1.6 Aim

The aim of this research is to deliver a new model and methodology for engineering scalable access control systems in organizations where hundreds or thousands of employees require access to information on a need to know basis in order to perform their job.

- The first objective is to challenge the long held belief, notion or sense that the number of subjects far exceeds the roles found in enterprise sys-

tems. If role explosion is a normal occurrence and previous models for the administration of RBAC consider role explosion a design problem this should motivate the requirement for role evolution and the introduction of new RBAC structural relationships that distinguish between roles and responsibilities.
- The second objective is to produce a new model for describing and reasoning about RBAC implementations using these new relationships to directly inform the architecture, thereby making RBAC systems more easily understood and receptive to change.
- The third and final objective is to deliver a new iterative methodology for engineering scalable access control systems, using the terms role and responsibility to improve communication.

This research has the potential to impact the perception of access control. Rather than being viewed as a necessary burden, access control has the potential to be a well understood, enabling technology, directly informed by the business model, a scaffolding for maintaining information systems, where the individual parts are simple to understand and the flexibility achieved through aggregating roles and responsibilities, provides the necessary framework for scalable access control systems.

1.7 Validation Approach

To the best of our knowledge, the current literature does not provide detailed research resulting in the delivery of a methodology for engineering scalable access control systems in organizations with hundreds or thousands of employees. The validation approach for this work is divided into three distinct phases.

First, after performing a literature review in chapter 2, we challenge the belief, notion or sense that the number of subjects far exceeds the roles found in IM systems in chapter 3 and chapter 4. We analyze the RBAC system found at ACME university, comparing it to a recently introduced fragment of RBAC called bi-sorted role-based access control [20]. We present our results and introduce a new role-centric methodology for dynamically constraining access to information. We describe how organizational scalability is enhanced at ACME university by decoupling subject and permission management at the expense of role evolution.

Next, in chapter 5 we produce our new model and deliver our new methodology for engineering scalable access control systems named ORGODEX. We use our first operational case study at ACME university to motivate the requirement for new RBAC structural relationships, distinguishing between roles and responsibilities. We advocate on behalf of role evolution.

Finally, in chapter 6, we validate our new model and methodology in our second operational case study. Assuming the duties of a Project Manager, we use our new ORGODEX methodology to lead a one year initiative to extend features and functionality of the ACME IM system to MECA, another educational institution. The project team is a distributed, multi-disciplinary and multi-lingual group of individuals with extensive to very little understanding of access control. We focus on the added benefits of harmonizing role and requirements engineering efforts when designing system functions and user interfaces [13].

1.8 Summary

In this chapter, we introduce our area of research, the authorization layer where security practitioners manage access to IM systems. When a subject successfully authenticates, for instance by correctly entering their user name and password, it is the authorization layer controlling what information they are authorized to read or write.

RBAC is a popular model for implementing the authorization layer, or WHO has access to WHAT, however there is no pervasive methodology used to produce scalable access control systems. This is an important concern because RBAC is an open, generic solution where there is no right or wrong answer but there is both an immediate and long term cost associated with its deployment.

In the following chapter, we perform a literature review, describing the work of authors who have influenced this dissertation. Then in chapter 3 we provide additional motivation for this work, introducing the notion of role explosion before advocating in favour of role evolution in chapter 4.

1.9 Publications

Publications related to this thesis:

- Aaron Elliott and Scott Knight. ORGODEX: Authorization as a Service. Proceedings of the 12th Annual IEEE International Systems Conference, to appear 2018.
- Aaron Elliott and Scott Knight. Start Here: Engineering Scalable Access Control Systems. Proceedings of the 21st ACM Symposium on Access Control Models and Technologies, pages 113-124, 2016.

- Aaron Elliott and Scott Knight. Towards Managed Role Explosion. In Proceedings of the 2015 New Security Paradigms Workshop (NSPW), number 1, pages 100-111, 2015. ACM Press.
- Aaron Elliott and Scott Knight. Role Explosion: Acknowledging the Problem. In Proceedings of the 2010 International Conference on Software Engineering Research and Practice, WORLDCOMP, pages 349-355, 2010. CSREA Press.
- Aaron Elliott and Scott Knight. One Employee and Several Applications: An Information Management Case Study. In Software Engineering Research and Practice, WORLDCOMP, pages 179-185, 2009. CSREA Press.

2 Literature Review

2.1 Introduction

In the previous chapter, we introduced the RBAC model for specifying the authorization layer and engineering access control systems. We also highlighted a deficiency for large organizations, identifying the requirement for a new methodology for engineering scalable access control systems that are well understood and directly reflect the business model of an organization.

In this chapter, we complete a literature review to place this work in context, highlighting the contributions of authors who have influenced this dissertation. In the following subsections, we chronologically order and present a series of models proposed for access control and its administration:

1. Role-Based Access Control (RBAC)
2. Administrative Role-Based Access Control (ARBAC)
3. Scoped Administrative Role-based Access Control (SARBAC)
4. Administrative Enterprise Role-based Access Control (A-ERBAC)
5. Attribute-Based Access Control (ABAC)
6. Relationship-Based Access Control (ReBAC)
7. Role-Centric Attribute Based Access Control (RABAC)
8. Responsibility MetaModel (ReMMo)

Finally, we conclude the chapter with a detailed description of Bi-Sorted Role-Based Access Control (RBÄC) an extension for existing RBAC implementations. The RBÄC model includes a return to first principles and the introduction of a new mandatory layer of abstraction for RBAC implementations intended to address scalability concerns.

2.2 Role-Based Access Control 1996 (RBAC96)

RBAC is the commercially dominant model for implementing the authorization layer or WHO has access to WHAT [39]. A role is a semantic construct around which access control policy is formulated. Roles are created for the various job functions in an organization and users are assigned roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. Permissions can be granted to roles as new applications and systems are incorporated and subsequently revoked as needed.

Unlike the concept of groups, which only specify a collection of users or subjects, roles identify a set of users and a related set of permissions. For this reason, RBAC administration is multi-faceted. User-Role Assignment (URA), Permission-Role Assignment (PRA) and Role-Role Assignment (RRA) are distinct actions required to bring users and permissions together. Role-role relationships are established to implement broad policy objectives [72].

The concept of RBAC began with multi-user and multi-application on-line systems in the 1970s. However, a resurgence of interest in the early 1990s was driven by a need for general-purpose features and the RBAC model was formally introduced by David Ferraiolo and Richard Kuhn at the 15th National Security Conference in October 1992 [26]. In 1996, Sandhu et al. introduced a novel framework of reference models, commonly known as RBAC96, to systematically address the diverse components of RBAC [72]. This includes the base model $RBAC_0$, $RBAC_1$ with role hierarchies, $RBAC_2$ with constraints and $RBAC_3$, the consolidated model. In the early 2000s, the National Institute of Standards and Technology (NIST) initiated a process to produce a standard, ANSI INCITS 359-2004, intended for software engineers designing products with RBAC [2]. This was followed eight years later with the revised version, ANSI INCITS 359-2012, maintaining the advantages of RBAC while providing a mechanism for including attributes in access-control decisions [3].

2.2.1 Base Model ($RBAC_0$)

The base model $RBAC_0$, identifies a collection of sessions and three entities; Users (U), Roles (R) and Permissions (P) where a user is a human being, a role is a job function or title and permissions are the approvals required to access one or more objects in a system. User Assignment (UA) and Permission Assignment (PA) are many-to-many relationships. A user can be a member of many roles and a role can have many users. Similarly, a role can hold several permissions, and the same permission can be assigned to several roles.

Each session is a mapping of one user to possibly many roles. The total permission set available is the union of all permissions granted to all the roles assigned and activated for the user in their session. A user may have multiple sessions open at the same time and each session may have a different combination of active roles. This feature of $RBAC_0$ facilitates the least-privilege principle, enabling administrators to keep user access to the minimum required for an individual to perform their duties [60].

Definition 1. The RBAC_0 model has the following components:

- U, R, P and S are sets of users, roles, permissions and sessions respectively
- PA $\subseteq$ P $\times$ R, a many-to-many permission to role assignment relation
- UA $\subseteq$ U $\times$ R, a many-to-many user to role enrollment relation
- user: S $\mapsto$ U, a function mapping each session, $\mathrm{s_i}$, to a single user, user($\mathrm{s_i}$), for a session's lifetime, and
- roles: S $\mapsto 2^{\mathrm{R}}$, a function mapping each session, $\mathrm{s_i}$, to a set of roles, roles($\mathrm{s_i}$) $\subseteq \{r|(user(s_i), r) \in UA\}$, which can change with time and session, $\mathrm{s_i}$, has the permissions, $\cup_{r\in roles(s_i)}\{p|(p,r) \in PA\}$

2.2.2 Role Hierarchies (RBAC_1)

The model RBAC_1 introduces Role Hierarchy (RH), a natural means for structuring roles to reflect an organization's lines of authority and responsibility. In a role hierarchy, senior roles acquire all of the permissions of junior roles. For example, in Figure 2.1, the Director inherits all of the permissions of both Project Leads 1 and 2.

Definition 2. The RBAC_1 model has the following components:

- user and the sets U, R, P, S, PA and UA are unchanged from RBAC_0
- RH $\subseteq$ R $\times$ R, is a partial order on R called the role hierarchy or dominance relation, also written as $\geq$, and
- roles: S $\mapsto 2^{\mathrm{R}}$ is modified from RBAC_0 to require roles($\mathrm{s_i}$) $\subseteq \{r|(\exists r\prime \geq r)[(users(s_i), r\prime) \in UA]\}$ which can change with time and session, $\mathrm{s_i}$, has the permissions, $\cup_{r\in roles(s_i)}\{p|(\exists r\prime\prime \leq r)[(p, r\prime\prime)] \in PA\}$

RBAC_1 acknowledges that it does not make sense for senior roles to inherit all of the permissions of junior roles. In such instances, RBAC_1 suggests the use of private roles or multiple inheritances. Using Project Lead 1 (PL1) and the Director (DIR) from Figure 2.1, for example, one could create a new role, PL1$'$, which is a new root node in the role graph senior to PL1. Using this method, one can assign permissions to PL1$'$ and grant PL1$'$ directly to Project Lead 1, thus blocking the implicit acquisition of roles by DIR.

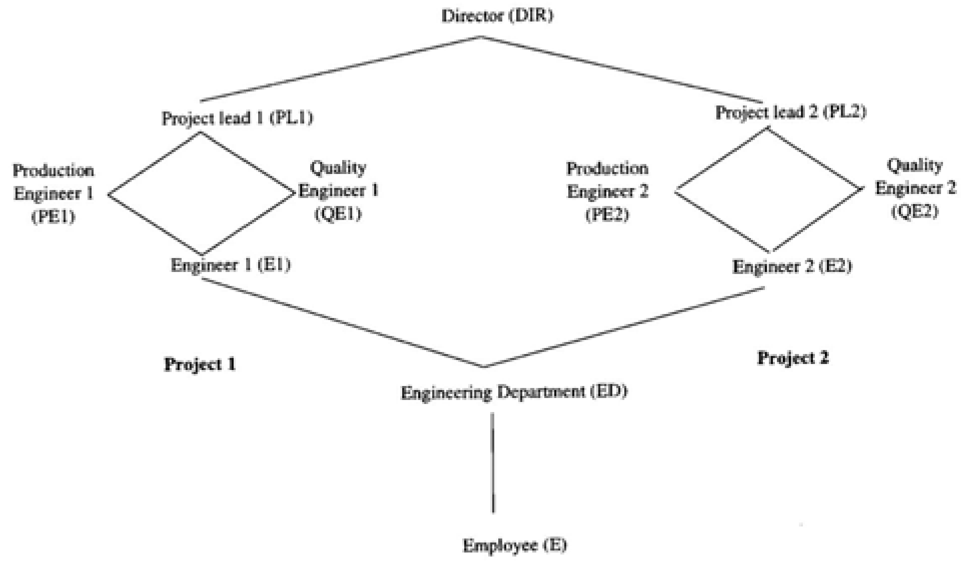

Figure 2.1: An example of a role hierarchy. Adapted from [74].

2.2.3 Constraints ($RBAC_2$)

The model $RBAC_2$ introduces the concept of Constraints (C). Sandhu et al. note that $RBAC_2$ is not a progression from $RBAC_1$ despite the labeling, both $RBAC_1$ and $RBAC_2$ are independent extensions of $RBAC_0$. Constraints are required to enable policies such as separation of duty where one member of an organization may not hold two roles simultaneously. The classic example is the purchasing and accounts payable manager positions in an organization whose duties are strategically segregated in order to avoid the possibility of fraud.

Definition 3. $RBAC_2$ is unchanged from $RBAC_0$ except for requiring that there be a collection of constraints that determine whether or not values of various components of $RBAC_0$ are acceptable. Only acceptable values are permitted.

Besides mutually exclusive roles supporting the separation of duties, another user assignment constraint is the number of members a role may have assigned to it. Similarly, the number of roles that any one user can be associated with could be limited. These are called cardinality constraints. Another frequently discussed constraint is the concept of prerequisite roles based on competency and appropriateness. For example, only persons with the employee role may be associated with the Engineering Department. It is important to note that constraints such as those discussed here assume a one-to-one correspondence between user identifiers and human beings.

2.2.4 Role Hierarchies and Constraints (RBAC$_3$)

A role hierarchy can be considered a type of constraint where all permissions assigned to a junior role must also be assigned to all senior roles. Sandhu et al. feel that it is appropriate to recognize the existence of role hierarchies in their own right in order to avoid redundancy of permission assignment and user assignment. RBAC$_3$ combines RBAC$_1$ and RBAC$_2$, providing both RH and C. However, there are several issues that arise when these two concepts are brought together. For instance, in Figure 2.1, the Director role inherits all of the permissions of the junior roles, Project Leads 1 and 2. If there is a constraint identified such that the Project Lead 1 and 2 are mutually exclusive then the Director role violates this mutual exclusion.

2.2.5 Administration

Despite the enthusiasm for RBAC, the use of RBAC principles to manage RBAC systems has been less widely studied [15]. A formalized model for the administration of RBAC should be both versatile and practical. Crampton and Loizou define *versatility* to mean that an access control model should be widely applicable and *practicality* to mean that an access control model could be implemented in a real-world system without incurring unacceptable overhead [16].

With RBAC, one can easily demonstrate practicality (Figure 1.4). By investing in role creation, one expects a return on investment for each and every User-Role Assignment (URA) that follows the first. Granting users access to information (or objects) is a well established computer science problem. The reference monitor, introduced by Anderson in 1972, models the controlled sharing of resources, validating all references made by a program in execution against those authorized for the subject by the system security policy [4]. Another basic abstraction used in dealing with access control is the access matrix introduced by Lampson in 1974. It describes a formal security model of protection state in computer systems, characterizing the rights of each subject with respect to every object in the system [49].

With RBAC, roles can be well understood by their names, and they determine the sets of permissions to be granted to users. In addition, it is easy to audit which users have access to a given permission and what permissions have been granted to a given user. A limited number of roles can represent many users or user types, and roles can be assigned to users by non-expert personnel [14].

Formal models for the administration of RBAC have been the subject of considerable research resulting in several important models. In the following subsections, the work of three influential sets of authors is further described.

2.3 Administrative Role-Based Access Control (ARBAC)

A literature review for the administration of RBAC typically begins with the Administrative Role-Based Access Control (ARBAC) family of models [74][71][63]. This includes the Administrative Role-Based Access Control 1997 (ARBAC97), followed by its extensions Administrative Role-Based Access Control 1999 (ARBAC99) and Administrative Role-Based Access Control 2002 (ARBAC02). The use of RBAC to manage RBAC remains a promising alternative for RBAC administration but the current models must be improved in order to achieve general acceptance. At the end of this chapter, the intuitions and assumptions inherent in the design of current ARBAC models will be summarized before the ramifications of these design decisions are analyzed in the subsequent chapter.

2.3.1 ARBAC 1997

ARBAC97, depicted in Figure 2.2, is based on the RBAC model and it describes the decentralized administration of User-Role Assignment (URA), Permission-Role Assignment (PRA) and Role-Role Assignment (RRA) [73]. In their introduction to ARBAC 1997, the authors offer the following commentary with respect to large enterprise systems:

- The number of roles can be in the hundreds or thousands and users in the tens or hundreds of thousands.
- Managing these roles, users and interrelationships is a formidable task that cannot realistically be centralized in a small team of system administrators.
- Decentralizing the details of RBAC administration without loosing central control over broad policy is a challenging goal for system designers and architects
- There is tension between the desire for scalability through decentralization and maintenance of tight control

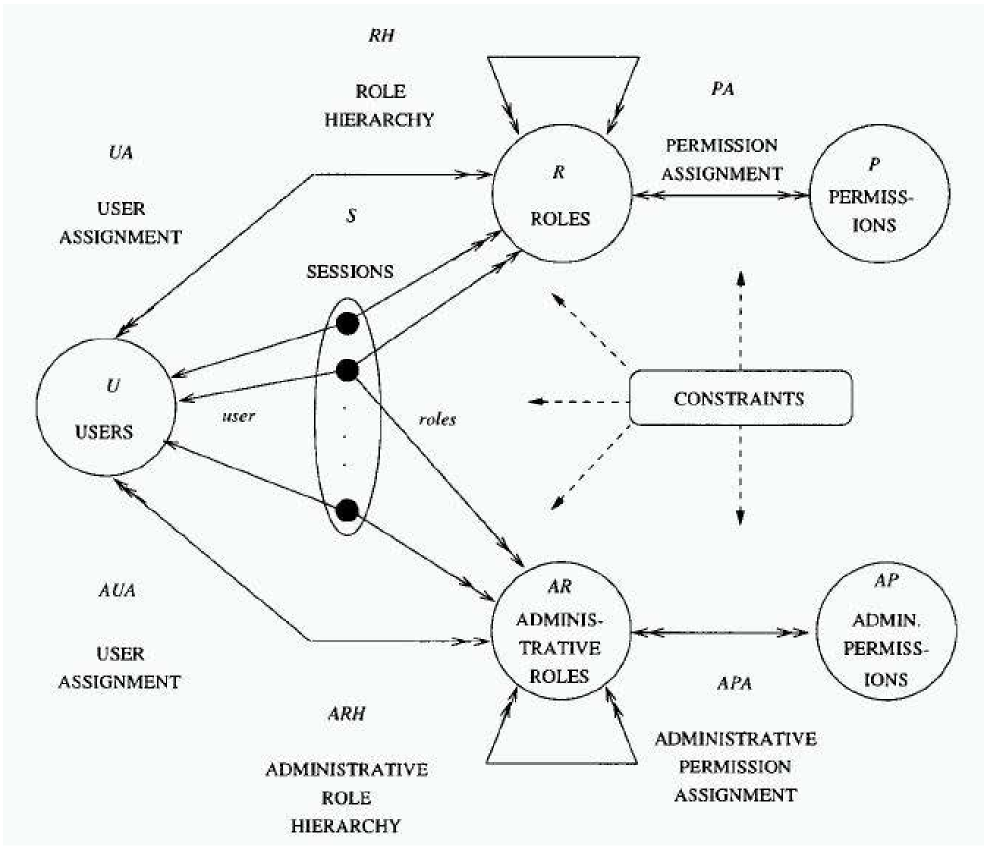

Figure 2.2: The ARBAC model [74].

- A complete solution to this problem requires further research and faces significant theoretical problems

In Figure 2.2, the bottom half is a mirror image of the top half and it is intended that Administrative Roles (AR) and Administrative Permissions (AP) be respectively disjoint from regular Roles (R) and Permissions (P). A single-headed arrow indicates a one-to-one relationship and a double-headed arrow describes a one-to-many relationship. Administrative permissions control actions such as adding new Users (U) and Roles (R), modifying User Assignment (UA) and Permission Assignment (PA) relations. Regular permissions, on the other hand, control operations on the data and resources and do not permit administrative operations. A user can be a member of many roles and a role can have many users. Similarly, a role can be assigned one or more permissions and permissions can be assigned to one or more roles. There is a partially ordered role hierarchy where x $\geq y$ signifies that role x inherits the permissions assigned to y and x is senior to y or equivalently y is junior to x. Inheritance along the role hierarchy is transitive and multiple inheritances are permitted for partial orders.

Each session in Figure 2.2 relates one user to possibly many roles. A user establishes a session and activates some subset of roles that he or she is a member of (directly or indirectly by means of a role hierarchy). The double-headed arrows from a session, S, to R and AR indicate that multiple roles and administrative roles can be simultaneously activated. The permissions available to the user are the union of permissions from all roles activated in that session. Each session is associated with a single user, as indicated by the single-headed arrow from the session to U. The concept of a session equates to the traditional notion of a subject in access control. A subject (or session) is a unit of access control, and a user may have multiple subjects (or sessions) with different permissions active at the same time.

User-Role Assignment (URA)

The aim of URA is to impose restrictions on which users can be added to a role and by whom, as well as to clearly separate the ability to add and remove users from other operations on the role. The notion of a prerequisite condition is a key part of URA. It enables administrators to explicitly define conditions where user-role relationships are not permitted. For instance, an administrator may explicitly define that a user, u, may not be granted the role, x, unless they have been previously granted the role, y.

Definition 4. *Prerequisite conditions* are boolean expressions using the usual $\wedge$ and $\vee$ operators on terms of the form x and $\bar{x}$ where x and $\bar{x}$ are regular roles (i.e. x $\in$ R). A prerequisite condition is evaluated for a user, u, by interpreting x to be true if $(\exists x\prime \geq x)(u, x\prime) \in UA$ and $\bar{x}$ to be true if $(\forall x\prime \geq x)(u, x\prime) \notin UA$. For a given set of roles, R, CR denotes the set of all possible prerequisite conditions that can be formed using the roles in R.

Definition 5. User-role assignments are controlled by can_assign $\subseteq AR \times CR \times 2^R$

The meaning of can_assign(x, y, {a, b, c}) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can assign a user whose current membership, or non-membership, in regular roles satisfies the prerequisite condition y to be a member of regular roles a, b, or c. URA uses the concept of role ranges to facilitate changes in the role hierarchy and add resiliency to the model. The notation in the following definition is based on the standard mathematical notation for open and closed intervals.

Definition 6. Role sets are specified in the URA model by the following range notation:

- $[x, y] = \{r \in R | x \geq r \wedge r \geq y\}$
- [x,y) $= \{r \in R | x \geq r \wedge r > y\}$
- (x,y] $= \{r \in R | x > r \wedge r \geq y\}$
- $(x, y) = \{r \in R | x > r \wedge r > y\}$

The URA revoke model is consistent with the philosophy of RBAC. In the URA model it is inconsequential that Bob has been assigned permissions by Alice. If Charlie replaces Alice in the organization and Bob is reassigned to another project then Charlie can revoke Bob's permissions regardless of the fact that Alice performed the original grant. This is a departure from the classical discretionary approach to revocation [75].

Definition 7. User-role revocations are controlled by can_revoke $\subseteq AR \times 2^R$

The meaning of can_revoke(x, Y) is that a member of the administrative role, x, (or a member of the administrative role that is senior to x) can revoke membership of a user from any regular role y/inY. Y specifies the *range of revocation* using the range notations previously defined.

The revocation operation in URA is considered weak because it only applies to the role that is directly revoked. Although RBAC permits a user to be an explicit and implicit member of the same role, weak revocation has an impact on the explicit membership only.

Definition 8. The user, u, is an explicit member of role x if $(u, x) \in UA$, and u is an implicit member of role x if for some $x\prime > x, (u, x\prime) \in UA$

Weak revocation only applies to explicit role memberships. If Alice has a session with the administrative roles defined by $A = \{a_1, a_2, ..., a_k\}$ and she tries to weakly revoke Bob from the role x then there are two cases to consider. In the first case, Bob is not an explicit member of x and the operation has no effect. In the second case, there exists a can_revoke tuple (b, Y) such that $a_i \in A, a_i \geq b$ and $x \in Y$ revokes Bob's explicit membership in x.

The algorithm for strong revocation is stated in terms of weak revocation and it applies to both explicit and implicit role memberships. If Alice has a session with the administrative roles $A = \{a_1, a_2, \ldots, a_k\}$ and Alice tries to strongly revoke Bob from role x then find all roles $y \geq x$ such that Bob is an explicit member of y. Weakly revoke Bob from all such y as if Alice performed this weak revoke in this session. If any of the weak revokes fail, then Alice's strong revoke has no effect, otherwise all weak revokes succeed. The algorithm for strong revocation cascades up a role hierarchy and it cascades down when Bob is explicitly revoked from a role b that is senior to a. However, if Bob is an explicit member of both b and a then Alice's revocation of b does not remove him from a.

Property 1. Implicit membership in a role a, is dependent on explicit membership in some senior role $b > a$. Therefore, when explicit membership of a user is revoked from b, implicit membership is also automatically revoked on junior role a, unless there is some other senior role $c > a$ in which the user continues to be an explicit member.

To conclude this summary of the URA model, Sandhu et al. note a lack of symmetry between the can_assign and can_revoke operations, in that can_assign involves prerequisite conditions and can_revoke does not. Enhancements to the range notation are left as future work.

Permission-Role Assignment (PRA)

PRA is concerned with permission-role assignment and revocation. From the perspective of a role, users and permissions have similar characteristics. They are essentially entities that are brought together by a role. Based on this notion, the authors propose that PRA is a dual of URA.

Definition 9. Permission-role assignment and revocation, respectively, are authorized in PRA by the following relations:

- can_assignp $\subseteq AR \times CR \times 2^R$
- can_revokep $\subseteq AR \times 2^R$

The meaning of can_assignp(x, y, Z) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can assign a permission whose current membership, or non-membership, in regular roles satisfies the prerequisite condition y to be a member of regular roles in range Z. The meaning of can_revokep(x, Y) is that a member of the administrative role x (or a member of the administrative role that is senior to x) can revoke membership of a permission from any regular role $y \in Y$.

Revocation for PRA is weak, so permissions may still be inherited after revocation. Like URA, strong revocation can be defined in terms of weak revocation for PRA, where the revocation of permissions cascades down (as opposed to up for the revocation of user membership).

Definition 10. Permission, p, is explicitly assigned to role x if $(p, x) \in PA$, and p is implicitly assigned to role x if for some $x\prime < x, (p, x\prime) \in PA$

Weak revocation only applies to explicit role assignments. If Alice has a session with the administrative roles defined by $A = \{a_1, a_2, ..., a_k\}$ and she tries to weakly revoke the permission, p, from the role, x, then there are two cases to consider. In the first case where p is not explicitly assigned to x, the operation has no effect. In the second case, there exists a can_revokep tuple (b, Y) such that $a_i \in A, a_i \geq b$ and $x \in Y$ revokes p's explicit assignment to x.

The algorithm for strong revocation is stated in terms of weak revocation and it applies to both explicit and implicit role memberships. If Alice has a session with the administrative roles defined by $A = \{a_1, a_2, ..., a_k\}$ and she tries to strongly revoke permission, p, from role x, find all roles $y \leq x$ such that p is explicitly assigned to y. Next, weakly revoke p from all such y. If any of the weak revokes fail, then Alice's strong revoke has no effect, otherwise all weak revokes succeed.

Role-Role Assignment (RRA)

RRA distinguishes three types of mutually disjoint roles; Abilities (A), Groups (G) and UP-roles (UP). Abilities are roles that can only have permissions and other abilities as members. Groups are roles that can only have users and other groups as members. UP-Roles are roles that have no restrictions on membership. Their membership can include permissions, users, groups, abilities and other UP-roles.

Abilities group the collection of permissions required to perform some task (e.g. open bank account) into a role. Abilities may be organized into hierarchies and assigning abilities to roles is similar to assigning permissions to roles. For this reason, the permission-role assignment model (PRA) is adapted to produce the Ability Role Assignment (ARA) model under RRA. Likewise, groups are a collection of users who are assigned as a single unit to a role (or team). Membership in the team may change over time and groups can be organized in a hierarchy. For group-role assignment, the authors adapt the user-role assignment model (URA) to define the Group Role Assignment (GRA) model under RRA.

Definition 11. Ability-role assignment and revocation, respectively, are authorized in ARA by the following relations:

- can_assigna $\subseteq AR \times CR \times 2^A$
- can_revokea $\subseteq AR \times 2^A$

Definition 12. Group-role assignment and revocation, respectively, are authorized in GRA by the following relations:

- can_assigng $\subseteq AR \times CR \times 2^G$
- can_revokeg $\subseteq AR \times 2^G$

Strengths and Weaknesses

The strength of the ARBAC 1997 model lies within the formality of its architecture and its direct linkage to the RBAC model. As the authors admit, a complete solution to the problem requires further research and faces significant theoretical problems. Several weaknesses have been identified with the ARBAC 1997 model, most notably, the unnecessary coupling of users with role hierarchies as described in [63][42][64]. In addition, the ARBAC 1997 model is administratively heavy, contains redundant user-role and permission-role information.

Recall Figure 2.1 and consider the following example [63]. A new Employee (E), John, is to become Quality Engineer #1 (QE1). To obtain the QE1 role, John must be a member of the Engineer #1 (E1) role, Engineering Department (ED) role and the Employee (E) role where there is a series of events that must occur in a predetermined order. This requires the coordination of two or more administrators as described in Table 2.1. In addition to the heavy administrative load, there is redundant user-role assignment information for John. As listed in Table 2.2, three of the records (i.e. 1, 2 and 3) result from the multi-step user assignment process. Since QE1 inherits E1, ED and E there is no longer a need for these user-role assignments.

In the chapters that follow, the need for a disjoint administrative hierarchy as prescribed by the ARBAC 1997 model is questioned. This work contends that security officers are a proxy for the people with the real authority. As a result, this unnecessarily adds to the administrative burden. By integrating with the Human Resources Management System (HRMS) as first described by Oh [63] organizations can empower their managers to do user-role assignment. A manager who has the authority to hire employees and delegate certain tasks should not have to ask an application administrator or security officer to delegate the task on their behalf.

Table 2.1: Administrative RBAC Actions

#	**Description**
1	Assignment of John to the role E
2	Assignment of John to the role ED
3	Assignment of John to the role E1
4	Assignment of John to the role QE1

Table 2.2: User-role Assignments

#	**Role**	**Assigned User**
1	E	John
2	ED	John
3	E1	John
4	QE1	John

2.3.2 ARBAC 1999

ARBAC99 extends ARBAC97 with enhancements for both User-Role Assignment (URA) and Permission-Role Assignment (PRA) and no change for RRA [73]. The important difference between ARBAC99 and ARBAC97 is the concept of mobile and immobile users and permissions. In URA, there are two consequences of assigning a user to a role. First, the user is authorized to use the permissions of that role and all of its juniors. Second, the user also becomes eligible for further role assignment by the appropriate administrative roles. These two aspects of role membership are tightly coupled in URA. The main innovation in ARBAC99 is the decoupling of these two aspects.

ARBAC99 distinguishes two kinds of membership in a role. Immobile membership grants the user the authority to use the permissions of the role but does not make that user eligible for further role assignment. The authors give several examples such as the case of a consultant who might be assigned to a project in the Engineering Department of XYZsoft and granted the role Project 1 Production Engineer (PE1) as an immobile member. Presumably, the role PE1 provides the consultant with the resources required to contribute to Project 1 and the immobile status prevents junior administrators from assigning the consultant to other Project 1 roles. Similarly, ARBAC99 distinguishes permissions that may not be delegated by assigning the immobile status. To formalize this distinction, one must consider that each role x consists of two sub-roles, Mx and IMx. Membership in Mx is mobile whereas membership in IMx is immobile.

Definition 13. For a given set of roles, R1, the corresponding set of roles in ARBAC99 are defined as $R = \{Mx, IMx | x \in R1\}$

Definition 14. There are four types of user-role memberships and four types of permission-role memberships in ARBAC99 for any given role x:

- Explicit mobile membership (EMx) for URA and PRA are defined as $u \in EMx \equiv (u, Mx) \in UA$ and $p \in EMx \equiv (p, Mx) \in PA$ respectively
- Explicit immobile membership (EIMx) for URA and PRA are defined as $u \in EIMx \equiv (u, IMx) \in UA$ and $p \in EIMx \equiv (p, IMx) \in PA$ respectively
- Implicit mobile membership (ImMx) for URA and PRA are defined as $u \in ImMx \equiv (\exists x\prime > x)(u, Mx\prime) \in UA$ and $p \in ImMx \equiv (\exists x\prime < x)(p, Mx\prime)(p, Mx) \in PA$ respectively
- Implicit immobile membership (ImIMx) for URA and PRA are defined as $u \in ImIMx \equiv (\exists x\prime > x)(u, IMx\prime) \in UA$ and $p \in ImIMx \equiv (\exists x\prime < x)(p, IMx\prime)(p, IMx) \in PA$ respectively

The strength of the ARBAC99 model is the formalization it shares with its predecessors; RBAC and ARBAC97. The weakness of the ARBAC99 model is that it contributes very little to the state of the art, in that, the solution provided is feasible but overly complicates the model while responding to the problem it is addressing.

2.3.3 ARBAC 2002

While acknowledging the elegance of ARBAC97, the authors of ARBAC02 illustrate redundancies and unnecessary couplings for both user and permission assignment in URA and PRA [63]. To address these practicality issues, ARBAC02 introduces the notion of organizational units as logical containers or *pools* for new users and permissions. The advantage of this methodology is that an ARBAC02 based implementation need not redundantly mirror organizational units and hierarchies already maintained in external systems and as a result, an ARBAC02 implementation is more conducive to changes in security policy.

An example of an ARBAC97 to ARBAC02 transform for the can_assign relation follows:

- ARBAC97: can_assign$(PSO1, E1 \wedge \overline{PE1}, [QE1, QE1])$
- ARBAC02: can_assign$(PSO1, @PJ1 \wedge \overline{PE1}, [QE1, QE1])$

Under ARBAC02, the @ symbol is a pointer to an organizational unit name in an external Human Resources Management System (HRMS) that *seamlessly* integrates with an ARBAC implementation. This addresses the issue of multi-step user-role assignment in ARBAC97 where new employee John must be assigned to the role Employee (E) followed by the role Engineering Department (ED) followed by the role Engineer #1 (E1) before the Project 1 Security Officer (PSO1) can assign John to the Quality Engineer #1 (QE1) role. Instead, ARBAC02 contends that the assignment of John to the organizational unit Project 1 (PJ1) in the HRMS places John @ the user pool where he may be assigned directly to Quality Engineer #1 (QE1) by the Project 1 Security Officer (PSO1). This eliminates the first three user-role assignments of ARBAC97, thus providing a seventy-five percent administrative savings.

The strength of ARBAC02 is its concept of user and permission pools. The weakness of ARBAC02 is its failure to quantify the administrative savings of this model. The authors determine that the functions of the Human Resources (HR) and Information Technology (IT) groups are outside the scope of role-based security administration. However, this model is entirely dependent on leveraging the use of an HRMS and the use of user and permission pools is not generic enough to support all systems [42].

2.4 Scoped Administrative Role-based Access Control (SARBAC)

In 2002, despite the enthusiasm for RBAC, the use of RBAC principles to manage RBAC systems had been less widely studied. Crampton and Loizou considered this a serious omission for dynamic access control systems where changes must be controlled, believing that it is best to develop a model for Role Hierarchy Administration (RHA) first since incorporating user-role and permission-role administration would be relatively easy to do later [16].

SARBAC extends the ARBAC model for administration, introducing *scoped* RHA to add flexibility and define domains where every role $r \in R$ has an administrative scope determined by the structure of the hierarchy. Informally, r' is in the administrative scope of r if any change to r' will only be observed by r and roles more senior than r [16]. More formally, administrative scope is defined as follows:

Definition 15. The administrative scope of a role r, denoted $\sigma(r)$, is defined to be $\sigma(r) = \{s \in {\downarrow} r : {\uparrow} s \subseteq {\updownarrow} r\}$.

The strict administrative scope of r is defined to be $\sigma(r) \setminus \{r\}$ and is denoted $\hat{\sigma}(r)$ implying that when $A \subseteq R$, $\sigma(A) = \{r \in \downarrow A : \uparrow r \subseteq \updownarrow A\}$ and $\hat{\sigma}(A) = \sigma(A) \setminus A$.

Crampton and Loizou establish a fundamental result, identifying that each pair of administrative domains is either nested or disjoint which leads naturally to the concept of administrative trees where the execution of a hierarchy operation will affect one or more roles through transitivity. As a consequence it may be necessary to *repair* the hierarchy relation following the addition or deletion of an edge in the role graph [16].

RHA_1 is the basic model and is applied directly to the role hierarchy. RHA_2 insists upon administrative permissions, offering finer granularity than RHA_1. RHA_3 introduces a binary relation where admin-authority $\subseteq R \times R$. If $(\alpha, r) \in$ admin-authority then α is called an *administrative role.* RHA_4 extends RHA_3 to control changes to the admin-authority relation.

Crampton and Loizo conclude that SARBAC is more complete, offering greater simplicity, practicality and versatility than the overly restrictive ARBAC97 model, detailing how SARBAC simplifies add operations and preserves administrative domains for delete operations [16]. With the admission that the simplicity of a model is a subjective quality, the authors state that structurally every SARBAC relation is simpler than its ARBAC97 counterpart. In terms of practicality and versatility RHA_4 is more *permissive* than ARBAC97 when considering thirteen specific role hierarchy operations where only two succeed under ARBAC97 and only two fail under SARBAC thereby raising concerns about the utility of ARBAC97.

The strength of SARBAC is its formalized approach to defining and maintaining administrative domains. Like ARBAC, the weakness of SARBAC is the practicality of using role hierarchies as a basis for defining administrative domains, with an all-powerful role at the root [51]. In addition, role hierarchies are not a natural approach for specifying the domain of administrators. Instead, administrative scopes are often defined based on organizational structures such as departments or cost centres [42]. Ultimately, both the ARBAC and SARBAC models lack of validation in a practical domain lead future researchers to *reconcile the requirements of the actual users of products implementing ARBAC with research* [42].

2.5 Administrative Enterprise Role-based Access Control (A-ERBAC)

In 2003, Kern, Schaad and Moffett arrived at similar conclusions. Like Crampton and Loizou, they identify the notion of *scopes* which describe the objects over which an administrator has authority [42]. However, unlike the authors of SARBAC, Kern et al. perform validations based on real-world experiences when implementing role-based administration infrastructures, critically evaluating and comparing their approach to existing models for administrative RBAC.

A-ERBAC describes the model employed by a commercial enterprise security management software solution deployed at a European bank where 70,000 users are administered [42]. Users are created and deleted automatically via a connection to the Human Resources Management System (HRMS) and the primary roles are assigned and revoked using automation. Kern et al. claim that Enterprise Roles (ER) are increasingly used by medium to large organizations as the basis for security management across different systems. In large RBAC implementations hundreds of administrators may be required to cope with the volume of users and their authority must be controlled by an internal mechanism, adhering to the principle of least privilege while supporting the creation and maintenance of roles.

A-ERBAC introduces the concept of *scopes* to control the authority of administrators on a Target System (TS). An administrator may be assigned the ability to view, insert, change or delete various RBAC elements such as users and roles provided that they are assigned one or more administrative *scopes* within the hierarchy of objects (e.g. Organizational Units or Cost Centres). Furthermore, Kern et al. argue that the *scopes* of A-ERBAC provide a more comprehensive solution than the *pools* of ARBAC02 as each *scope* is optionally associated with attributes that enrich the administrative convenience.

In the A-ERBAC model, scopes are not limited to the organizational hierarchy because other objects are often relevant in the enterprise. In the experiences provided, the authors observed cases where the administration of a TS was performed by different departments and in one instance, a bank made a clear distinction between system security (e.g. operating system access control) and application security.

The strengths of A-ERBAC follow from the strong practical background of the authors presenting this work and the introduction of scopes which is more generic and flexible than the use of pools in ARBAC02. Additionally, the concept of ER is an important step towards the consolidation of role information, describing a central repository where applications can integrate a shared set of roles instead of each redefining novel sets.

The A-ERBAC model describes a second layer of roles between subjects and permissions without highlighting the fact that this implies a minimum of two roles for each user (i.e. subject-role-role-permission) where new semantics have been informally introduced, differentiating business and functional

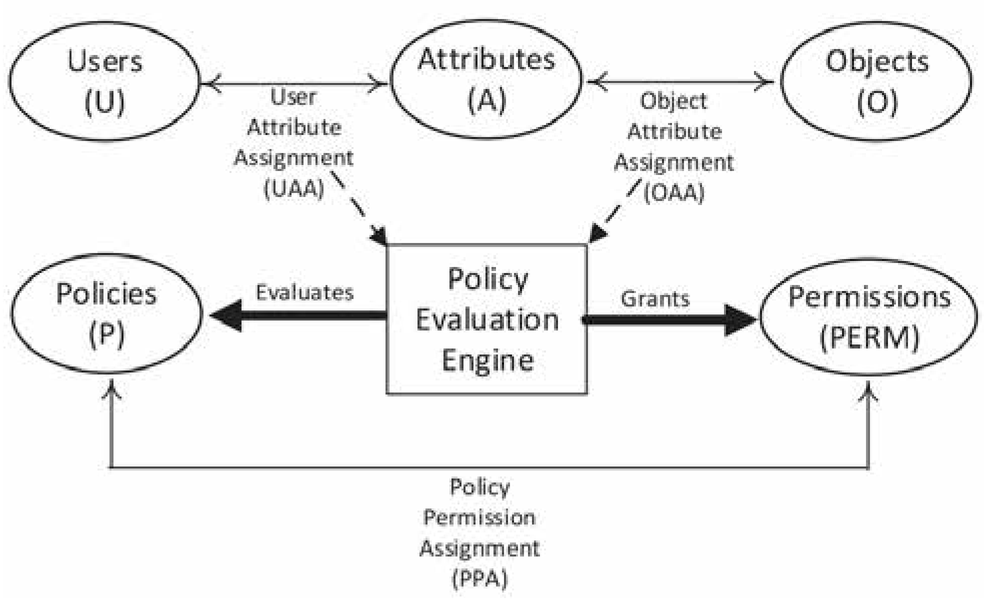

Figure 2.3: The core ABAC model . Thin solid arrows denote many-to-many relations, thick solid lines denote relation with policy engine, and dotted lines denote information used by the policy engine to evaluate a given policy. Ovals represent ABAC model elements [78].

roles. While we agree that the scopes of A-ERBAC are an important addition, the notion of typed roles might be equally important despite its unheralded presentation in this work.

2.6 Attribute-Based Access Control (ABAC)

Several recent papers related to access control focus on the debate between ABAC and RBAC. According to proponents of ABAC it is newer, simpler to implement, and accommodates real-time environmental states as access control parameters [14]. ABAC asserts that access control can be determined based on various attributes presented by a subject [40]. In its most basic form, ABAC relies upon the evaluation of attributes of the subject and object, environmental conditions and the access control policy defining the allowable operations for the subject-object attribute combinations [34].

With reference to Figure 2.3, each object within a system must be assigned specific attributes, characterizing the object. Likewise, each subject that uses the system must be assigned specific attributes. The policy evaluation engine necessitates every object within the system to have a least one policy defining the access rules for the allowable subjects, operations and the environmental conditions. Once attributes and policies are established, objects can be protected using ABAC. The policies that may be implemented in an ABAC model are only limited to the degree imposed by the computational language and the richness of the available attributes.

There are several open problems with ABAC primarily stemming from the increased complexity introduced when attempting to enhance the flexibility and generality of access control policies [78][34][39]. The lack of an agreed upon or foundational model and limited emulation for representing traditional models does not facilitate a transition from legacy RBAC implementations to ABAC. Moreover, there are concerns about hierarchical ABAC, auditability, formal security analysis, separation of duties, delegation, attribute storage and sharing between different organizations, scalability, administration and user comprehension.

RBAC and ABAC have their particular advantages and disadvantages. ABAC may require up to 2^n rules for n attributes and RBAC could require 2^n roles in the worst case. In essence, RBAC trades up-front role engineering effort for ease of administration and auditing, conversely ABAC may require less up-front engineering effort at the expense of complicating user audits [45][14].

For example, if a subject is currently on project A but also sometimes works on projects B and C, rules must be instantiated and evaluated for each of their projects. If the subject has another attribute with three possible values (1, 2, or 3), then rules must be instantiated with nine possible value combinations for these two attributes, quickly reaching a combinatorial explosion of possible rule instantiations to evaluate.

The major challenge of ABAC is the just-in-time evaluation of its rules making it extremely difficult, if not impossible, to determine the permissions available to a particular user [23]. On these grounds alone we consider ABAC insufficient as a security model. With ABAC how does one report on the *protection state* of their computer system when breaches inevitably occur [49][31]. We strongly support a hybrid RBAC-ABAC approach similar to what is described in section 2.8, introducing our new hybrid model in section 4.4.

2.7 Relationship-Based Access Control (ReBAC)

In many emerging application domains, such as health care and education, it is more natural to determine authorization decisions based on a relationship (e.g. doctor-patient) thus identifying the requirement for access control systems to support the sharing of wide-scope relationships across multiple access contexts and to do so in a rational manner [27]. Social Network Systems (SNS) explicitly track the social networks of their users, for example Facebook permits one individual to invite another to connect. If the invitation is accepted, the relationship is confirmed and a new edge in the social graph is created between two friends. Gates first described a new paradigm for access control based on interpersonal relationships, highlighting the requirement for access controls that are (1) relationship-based, (2) fine-grained, and (3) support interoperability [28].

ReBAC is formulated to capture the core idea of employing social networks as the basis of authorization decisions, enabling the explicit tracking of interpersonal relationships between users, and the expression of access control policies based on these relationships. Contrary to other SNSs, ReBAC relates several notional connections (e.g. child-parent and doctor-patient) with directional information (e.g. child-parent and parent-child are distinct). These features allow the model to capture rich domain concepts where relationships and authorizations are articulated in terms of context [27].

ReBAC is not entirely distinct from RBAC and may be considered a natural generalization through the use of binary relations over users rather than roles for capturing domain knowledge. Fong believes RBAC has been pushed to the limit to cope with this demand, demonstrating why previous work such as *role templates* is merely a way of encoding a binary relationship, for example manager(john) [29]. There exists a number of parallels between RBAC and ReBAC including the following:

- user-role enrollment and inter-user relationships
- permission-role assignment and policies
- sessions and contexts
- role and context hierarchies
- separation of duty constraints and well-formed contexts

The health care domain is described as the archetypal application realm where ReBAC is most required. For example, in support of security policy statements like *my patient record should only be accessible to my family doctor, not all doctors*. In [69], the authors demonstrate the feasibility of ReBAC, incorporating this model into a production-scale medical records system (i.e. OpenMRS) with backward compatibility for the legacy RBAC mechanism. This is an important accomplishment however the performance evaluation indicates that the implementation is not yet deployed in any clinical setting, therefore no production data set is available and synthetic data is utilized instead. In addition the authors report unacceptable authorization times when relying on a relational database implementation.

Rizvi and Fong extend ReBAC to support fine-grained interoperability between the ReBAC model and legacy RBAC models, correctly identifying that due to significant investment, RBAC is not going to disappear any time soon, stressing that any extension of an existing software application to incorporate ReBAC must find a way for the new access control model to integrate harmoniously with the legacy RBAC model [68]. With this objective the authors introduce the notion of demarcations and demonstrate how RBAC and ReBAC can interoperate in interesting ways. Referencing the work of [46], they recognize the value of creating demarcations as abstract groupings of privileges decoupled from user management.

We agree that demarcations are an important notion and we describe the work of Kuijper and Ermolaev in detail in section 2.10. However, in this dissertation we propose a way forward for access control and information security that is **not** based on new ideas like ReBAC and ABAC whose implementation is essentially orthogonal to RBAC [68]. Instead, we propose extensions for RBAC that offer the advantages championed by these alternative models for access control.

2.8 Role-Centric Attribute Based Access Control (RABAC)

It has long been recognized that traditional RBAC formulations are inefficient when fine-grained access control is required, for instance, the familiar doctor-patient problem where a doctor is only allowed to view the records of his own patients. Under the NIST RBAC model a doctor role needs to be defined for each patient, thus dramatically increasing the number of roles. Anecdotally, current practice indicates that organizations work around these limitations in ad-hoc ways.

In 2010, NIST announced an initiative to unify and standardize various RBAC extensions by integrating roles with attributes, identifying three alternatives [45]:

1. Dynamic Roles. The first option employs user and context attributes to dynamically assign roles to users.
2. Attribute Centric. In this option roles are simply another attribute of users.
3. Role Centric. The third option proposes that the activated set of roles for a user session be reduced based upon attributes.

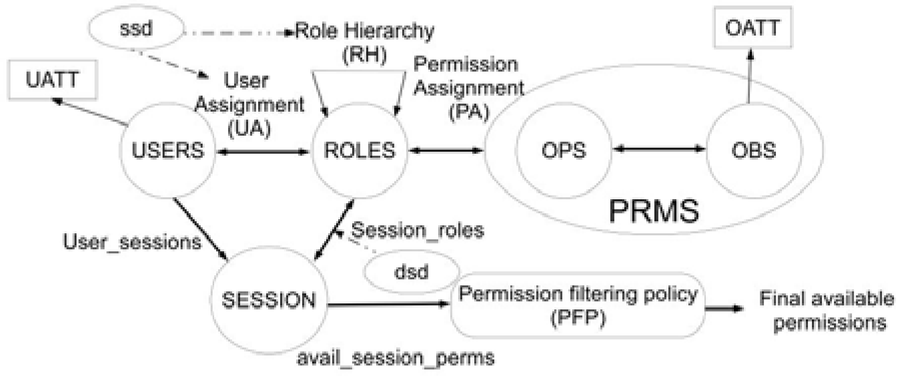

Figure 2.4: The RABAC model extends the NIST RBAC model, adding user and object attributes combined with permission filtering policies [39].

Combining the benefits of RBAC and ABAC to incorporate the advantages of each model is an interesting approach fueled by the ongoing RBAC versus ABAC debate. With RABAC, Jin and Sandhu propose a novel role-centric attribute-based access control model as an extension to the RBAC model, offering a path for practical deployment based on the addition of User Attributes (UATT), Object Attributes (OATT) and a Permission Filtering Policy (PFP) as depicted in Figure 2.4 [39]. Attributes are either atomic or set-valued and the PFP constrains the available set of permissions using filter functions $\{F_1, F_2, F_3 \ldots F_n\}$ to return a boolean expression based on user and object attributes. The result is a set of Permissions (PRMS) including support for both static (ssd) and dynamic (dsd) separation of duties, explained briefly in section 7.6.

ABAC and RBAC may be combined judiciously to offer access control that is scalable, flexible, auditable and understandable [14]. RABAC is the first model to integrate attributes using a role-centric approach thereby maintaining static relationships between roles and permissions to facilitate the determination of risk exposure [44]. This model is important because it addresses the *undefined* role explosion problem with an extension to the commercially popular and mature RBAC model, unlike ABAC and ReBAC. However, RABAC does not semantically separate roles and responsibilities like our model. In addition, we present role versus user attributes as a preferred basis for engineering PFPs.

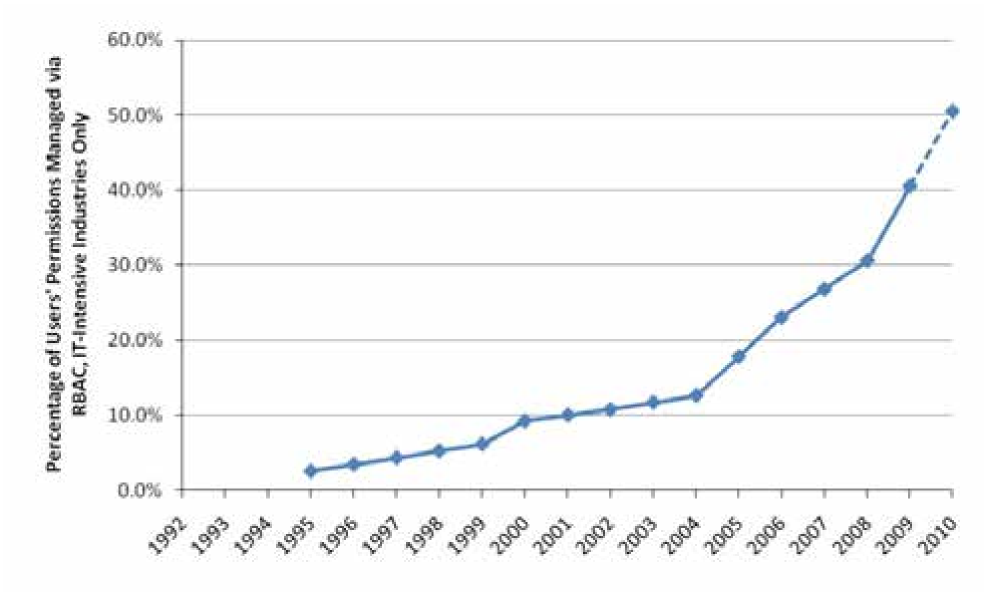

Figure 2.5: RBAC adoption between 1992-2010 [58].

2.9 Responsibility MetaModel (ReMMo)

According to O'Connor and Loomis [58], the use of RBAC has grown steadily since 1994, with the rate of adoption accelerating in both 2004 and 2008 as visualized in Figure 2.5. Their analysis estimates that between 1994 and 2009, the use of RBAC yielded a net savings of 6 billion dollars for American businesses. Furthermore, their study indicates that the number of employees whose permissions are at least partially managed by roles grew from only 2.5% in 1995 to 40.5% in 2009 and they estimate this number will grow to 50.5% of users at organizations with more than 500 employees by 2010.

With the success and rapid adoption for RBAC, Feltus stresses the requirement for engineers to *define methods and techniques* to align information systems with the business processes they support [25], observing the following three critical problems when modeling and engineering employee access rights:

- Insufficient analysis of the business roles as defined in the employment contract or job profile
- Misalignment between the business roles and those defined at the application layer
- Misalignment between the employees responsibilities and their access rights

With the objective to better align business and IT, the ReMMo is proposed as an enhancement to RBAC built around three concepts. The first concept defines the obligation of the employee to perform an action. The second concept describes the rights required by the employees to fulfill an obligation and the third concept identifies the assignment process as the *the action of linking an employee to a responsibility* [24].

The ReMMo proposes the introduction of responsibilities as an intermediary concept between the user and the role in RBAC. Feltus et al. suggest that employees should be assigned specific responsibilities independent of roles and that permissions are associated with the responsibilities for which they are required. This model allows them to refine the URA concept of RBAC where users are assigned to responsibilities as far as they commit to them. The responsibility is an abstract concept that could be either a concrete atomic responsibility or a group of responsibilities. The PRA concept of RBAC is refined through associating permissions both to atomic responsibilities and to roles.

The tuple of concepts [user-role-responsibility] facilitates defining two types of user-role assignments and one type of responsibility-role assignment:

1. Direct role assignment: an employee is assigned to a role and gets the corresponding responsibilities and permissions. In that case, the role is often the main function of the employee and corresponds to his main function in the company.
2. Direct atomic responsibility assignment: An employee is assigned an atomic responsibility without any associated role and the employee then gets the corresponding permissions.
3. Indirect role assignment: an employee is assigned, by direct atomic responsibility assignment all the responsibilities that compose a predefined role, so he is implicitly assigned to the role and he gets the permissions corresponding to those responsibilities. This case reflects the situation where an employee is assigned to more and more responsibilities which happens to be the responsibilities predefined in a role. Whereas from an IT point of view, the set of these responsibilities correspond to a role, the employee does not have the title corresponding to the role, from an organizational viewpoint.

The strength of the ReMMo includes its notion of independently modeled responsibilities combined with its objective to better align business and IT. However, we feel it is a precarious proposition to advocate the performance of *direct atomic responsibility assignment* in organizations with high volumes of employee turnover [21].

In the following section, we introduce RBÄC, a new principled approach to engineering RBAC that does not permit the assignment of responsibilities directly to users. We absolutely agree with the authors of this model and to ease comprehension for the reader they should understand the notion of *demarcations* is equivalent to the concept of responsibilities described in this section.

2.10 Bi-Sorted Role-Based Access Control (RBÄC)

RBÄC is presented as a fragment of RBAC which may be applied to existing RBAC implementations, the perceived added value lies within the conceptual boundaries it introduces, decoupling subject and permission management, thus introducing a higher administrative level for access management [46]. For practitioners, this decoupling implies that modeling (1) subjects and (2) permissions is broken into independent activities. With these two aspects maintained by suitable teams, security officers may configure access control rules at an appropriate level of abstraction.

In addition, RBÄC inherently facilitates many-to-many administrative mutations and ultimately leads to more organizational scalability. The speculation being that such an approach might prove beneficial in the following senses:

- subject management can be delegated as appropriate in organizations, reducing administrative overhead.
- application architects can focus on creating independent roles based on the functional requirements.
- security officers can perform access management at an appropriate level of abstraction.

RBÄC describes two distinct objects of indirection: the (1) proper role and (2) demarcation which are used to distinguish conceptual boundaries for the administration of RBAC [46]. This is not a new idea. Kuijper and Ermolaev cite Oh and Park as the first researchers proposing permissions be grouped independently into *task-based roles* [62]. Next, the work of Kern et al. [42], on Enterprise Role-based Access Control (ERBAC) is provided as an example, clearly demonstrating the practicality of maintaining two distinct role hierarchies. Finally, the work of Nyanchama and Osborn [57] is referenced as further evidence that a dichotomy exists between subject and permission

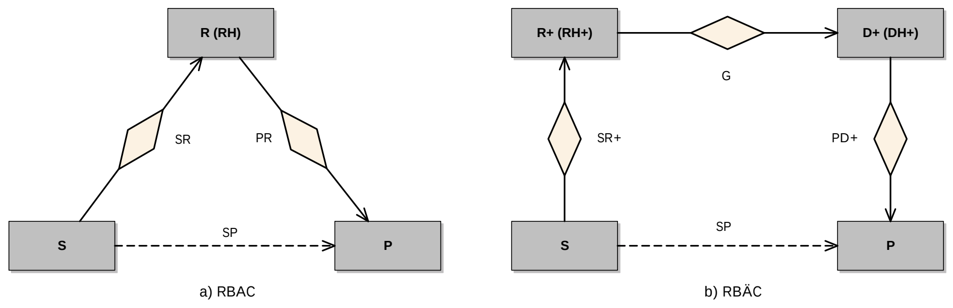

Figure 2.6: RBÄC proposes a new conceptual boundary between subjects and permissions. Adapted from [46].

management. The important difference to consider with RBÄC is the assertion subjects and permissions are never linked by a single role. Instead, there is **always** *at least two roles* between a subject and a permission.

Unlike previous extensions to the classic RBAC model [72], RBÄC revisits first principles with the hypothesis that organizational scalability is facilitated when permissions are managed independently from subjects. RBÄC is presented as a *fragment* of RBAC, a conceptual shift, from the *triangular* RBAC model to the *desirable square* model of RBÄC. In Figure 2.6a, we see that RBAC introduced the role as a layer of indirection between subjects and permissions. SP is the implicit result of assigning permissions to a role (PR) and then enrolling subjects into this role (SR). To achieve the square of RBÄC in Figure 2.6b, another layer of indirection is introduced with roles being categorized as either a proper role (R^+) or a demarcation (D^+) . Permissions are assigned to demarcations (PD^+) and subjects are enrolled into proper roles (SR^+). Permissions are never assigned to proper roles directly. Instead all subjects obtain permissions indirectly through the grant relation (G) where proper roles and demarcations are *linked up*. Figure 2.6 slightly adapts the work of Kuijper and Ermolaev, adding directional arrows to depict the explicit role hierarchy that exists between subjects and permissions.

Definition 16. RBÄC retains the principal semantic domains underlying RBAC (i.e. S and P) and defines the following syntax:

- Let S be a set of subjects
- Let P be a set of permissions
- Let R^+ be a set of proper roles
- Let D^+ be a set of demarcations [1]

[1] R^+ and D^+ are disjoint sets

- Let $SR^+ \subseteq S \times R^+$ be a subject-proper-role assignment relation
- Let $PD^+ \subseteq P \times D^+$ be a permission-demarcation assignment relation
- Let $RH^+ \subseteq R^+ \times R^+$ be a proper-role-hierarchy relation, RH^+ is required to be acyclic
- Let $DH^+ \subseteq D^+ \times D^+$ be a demarcation-hierarchy relation, DH^+ is required to be acyclic
- Let $G \subseteq R^+ \times D^+$ be a grant relation

Definition 17. The semantics of RBÄC identify the access relations SP $\subseteq S \times P$ such that $(s,p) \in SP$ iff there exists roles $r, r' \in R^+$ and demarcations $d, d' \in D^+$ and the following conditions hold:

1. $(s,r) \in SR^+$, i.e.: subject s is a member of proper role r.
2. $r \geq_r^+ r'$, i.e.: $r = r'$ or r is a senior role of r' [2]
3. $(r',d') \in G$, i.e.: proper role r' is granted access to demarcation d'
4. $d' \geq_r^+ d$, i.e.: $d = d'$ or d is a sub-demarcation of d'. [3]
5. $(p,d) \in PD^+$, i.e. permission p is part of demarcation d

For small organizations where the number of roles remain relatively few, classic RBAC is often an adequate solution. However, for large organizations where there is an ongoing requirement to support employee turnover, policy changes and reorganization, RBÄC is a logical evolution. Despite the advantages of RBAC, the administrative degrees of freedom become limited when practitioners utilize a triangular RBAC model, Figure 2.6a), where there are four basic mutations:

1. Enroll a subject $s \in S$ to role $r \in R$, i.e.: add (s,r) to SR
2. Disenroll a subject $s \in S$ from role $r \in R$, i.e.: remove (s,r) from SR
3. Assign a permission $p \in P$ to role $r \in R$, i.e.: add (p,r) to PR
4. Unssign a permission $p \in P$ from role $r \in R$, i.e.: remove (p,r) from PR

The effect of an atomic RBAC mutation on SP is always one-to-many or many-to-one and never many-to-many. This is referred to as the administrative micro-stepping problem [46]. With RBÄC the intent is to *break away* from the classic triangular example of RBAC, enforcing another degree of freedom and facilitating many-to-many mutations for SP. For practitioners this additional layer of abstraction permits administrative degrees of freedom not enjoyed under the classic triangular RBAC model.

As previously, stated this is not a new idea and one could certainly design RBAC systems in this fashion prior to RBÄC. However, with this fragment

[2] $\geq_r^+$ defines the transitive reflexive closure of RH^+
[3] $\geq_d^+$ defines the transitive reflexive closure of DH^+

of RBAC the suggestion is that one would **never** assign permissions to a role granted directly to users. In the strictest sense a subject would never receive a permission in a subject-role-permission mapping. This guarantees that every subject has at least two roles and often many more.

RBÄC is explained in the context of a physical access control system. In chapter 4, we analyze the RBAC implementation used by ACME university to secure its student information system where access is granted on a *need to know* basis. In extreme cases subjects have more than one hundred roles. Many of these roles are demarcations or discrete units of functionality delivered organically over time as functional or task-based roles. Readers who are familiar with this area of research may be experiencing a strong sense of disbelief at this point. Others might immediately decide this is a poorly designed RBAC system. Ten or more years ago we might have been equally critical but today we know that ACME university is not the only organization describing subjects with more than one hundred roles [38].

2.11 Summary

In this chapter, we complete a literature review to place this work in context, highlighting the contributions of authors who have influenced this dissertation. We begin with a review of RBAC before reviewing three models for the administration of RBAC including ARBAC, SARBAC and A-ERBAC. Then we review access control models proposed to address the perceived weaknesses of RBAC. We provide a chronology for the maturation process of RBAC and its administration before reviewing alternative solutions and considering their strengths and weaknesses.

Finally, we conclude this chapter with a detailed description of RBÄC, a new fragment of RBAC and a return to first principles, resulting in the introduction of a new mandatory layer of abstraction guaranteeing the desirable square subject-role-role-permission relationship. In the following chapter we provide additional motivation for this work, introducing the notion of role explosion before advocating in favour of our new concept termed role evolution.

3 Role Explosion

3.1 Introduction

In the previous chapter we reviewed papers for RBAC and its administration, explored alternative models and concluded with a relatively new RBAC model insisting upon two mandatory layers of roles between subjects and permissions.

In this chapter, we introduce the term *role explosion*, conducting a real-world case study for one employee in the Government of Canada (GoC) to better appreciate this concern and its symptoms. We provide motivation for satisfying our first objective, challenging the long held belief, notion or sense that the number of subjects far exceeds the roles found in enterprise systems.

First, we provide background information for our case study. Next, we propose two symptoms of role explosion. Then we perform an empirical study, investigating and analyzing the role information related to one employee of the GoC. Finally, we discuss our results and present our conclusions.

At the conclusion of this chapter, the reader will better appreciate why role explosion occurs in large organizations where *more and more new types of applications that require controlled sharing of resources or discrimination of information appear* [9]. If role explosion is a normal occurrence and previous models for the administration of RBAC consider role explosion a design problem this should motivate the requirement for new RBAC structural relationships.

3.2 Background

During the last few decades, the use of RBAC has exploded within organizations such as the GoC. As employees enter and exit large GoC organizations there are a number of security challenges present when on-boarding and off-boarding employee access to IM systems. Our research demonstrates that new GoC employees may be granted access to several applications [18]. In large organizations where employees are constantly entering, exiting and moving between organizational units, it is extremely challenging to maintain access control systems that are tightly coupled with security policy. The Government of Canada Security Policy (GoC-SP) states that department's *interoperability and information exchange are enabled through effective and consistent security and identity management practices* [60].

IM systems provide access to corporate information in dynamic, heterogeneous infrastructures with challenging combinations of employee turnover and security policy. Protecting information and client data is an on-going concern for large public and private organizations. In the GoC, one of these challenges is the least-privilege principle. Under section 16.4.3 of the Operational Security Standard: Management of Information Technology Security (MITS), departments must keep access to the minimum required for individuals to perform their duties (i.e. the least-privilege principle), and ensure that they are regularly updated to accurately reflect the current responsibilities of the individual [59].

In its policy statement, the GoC defines IM as *a discipline that directs and supports effective and efficient management of information in an organization, from planning and systems development to disposal or long-term preservation* [61]. An IM system can be a paper-based filing cabinet under lock and key or it can be a large centralized database. In the latter case, employees might retrieve information using an application developed to permit access to corporate information in dynamic, complex environments. After a user successfully enters their user name and password, authenticating their identity, an application may limit their access to information.

The on-boarding process for a new employee in the GoC may include a series of administrative actions that grant access to one or more applications. Likewise, the off-boarding process for a departing employee in the GoC may include a series of administrative actions that remove access from one or more applications. Collectively, these processes are elements of a larger business process referred to as *employee turnover*.

An example for one microscopic aspect of an RBAC implementation is pictured in Figure 3.1 where the employee turnover process is occurring, old-employee-out (1) and new-employee-in (2). Each user account is enrolled into one or more roles and each role is assigned various permissions. In this example, lines connecting a role to a folder imply that a role has the authorization required to both read and write information to a database. Lines from a role to a grid indicate that the role is assigned permissions to read information.

This example describes access to one application for five different employees where twelve roles have been implemented to restrict read and write

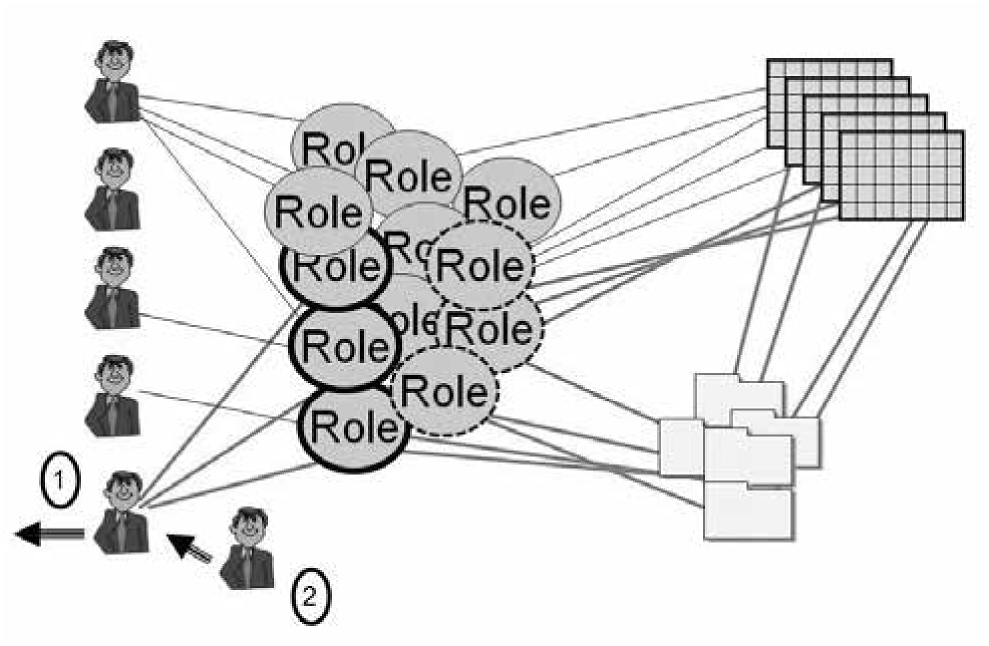

Figure 3.1: Old-employee-out (1) and New-employee-in (2).

access to information. Several administrative actions are required to revoke permissions from a departing employee and grant access to a new hire. Pause and consider that this example relates to one application and employees often require access to several applications in large organizations. Managing and monitoring who has access to what, on an individual user and resource basis, can be very costly for large organizations. As a result, it is very challenging to maintain a practical RBAC implementation that is tightly coupled with an organization's security policy.

In this chapter the term *role explosion* is symptomized and supported by our real-world case study. Results are presented for one employee in the GoC and the advantages of on-boarding and off-boarding by staffing position are discussed. As the number of applications for each employee increase, so too does the cost of the related administrative processes.

3.3 Symptoms

To our knowledge the term role explosion has not been formally defined by the research community. Instead it would seem there is a belief, notion or sense that some threshold ratio of roles to subjects indicates bad RBAC design [45][39][34]. We propose the following two symptoms of role explosion:

- **Symptom 1** An organization requires employees to access several applications autonomously managing their own set of role information.
- **Symptom 2** An organization has one or more applications where the ratio of roles to users approaches or exceeds 1.

Symptom 1 contributes to the role explosion problem because it is administratively costly to introduce and maintain redundant role information across several applications.

Symptom 2 contributes to the role explosion problem because it is administratively costly to maintain an application with an excessive number of roles.

At the RBAC2000 Workshop, the number of roles was estimated to be 3-4% of the subject population [76]. We are curious if oral estimates have been provided at similar workshops over the past two decades and we wonder whether this percentage holds today. Several models have been proposed for maintaining access control implementations. Proofs for the utility of these models are typically restricted to contrived examples that fail to reflect the complexity of medium to large organizations [76].

Organizations with extensive work breakdown structures organically move towards role explosion over time as more and more applications are integrated into the business infrastructure and more and more specialization (or customization) is supported in software features and functionalities. Table 3.1 lists role and subject information for a real-world application by calendar year. The role granularity metric is a simple ratio, comparing the total number of roles and subjects.

The data shows a ratio higher than 1 from the onset, providing real-world evidence for symptom 2 of role explosion at the macro level. In 2008, we have an example of an application where the actual number of roles is 150% of the subject population. This explosive role growth speaks to the popularity of RBAC but it does not support the 3-4% estimate provided at the RBAC2000 Workshop.

At the micro level, we are curious why this explosive growth is occurring and we wonder how it is being managed. In the following subsection, we perform an empirical study to better understand why this might occur.

Table 3.1: Role and User Information for a Representative Application

Year	**Roles**	**Subjects**	**Roles/Subject**
2005	348	271	1.3
2006	441	295	1.5
2007	506	335	1.5
2008	563	379	1.5

3.4 Empirical Study

The following case study is an investigation and analysis of the role information related to one employee in the GoC. Tables are used to summarize the research results. The participant is Alice, the Administration Officer for the Information Services department. The responsibilities of Alice's staffing position dictate that she is granted access to the 7 applications listed in Table 3.2 and the roles associated with each of Alice's accounts are listed in Table 3.3.

This IM case study is an investigation and analysis of the on-boarding and off-boarding procedures for one GoC employee and several applications with the following constraints:

- Each application is required by the employee to complete the tasks associated with their staffing position
- Each application requires authentication, for instance the successful entry of a user name and password

If one GoC employee requires 7 different applications and each account (i.e. user name and password) is enrolled into 2 roles, for example, the number of roles being managed for one employee is 14. If GoC employees are requiring access to more and more secure applications and the number of roles created to limit (or customize) access for each application is increasing, one might conclude that the administration of RBAC is a growing concern.

- 1 GoC Employee
- 7 Applications or Services / Employee
- 2 Roles / Application / Employee

As new employees enter the GoC, there is an associated cost with managing the on-boarding process and this directly relates to the maturity of human resource relationships like those found in this study. If Alice leaves the organization before the new employee arrives and Alice's application requirements have not been thoroughly documented then the on-boarding process may prove

Table 3.2: Applications Required by our Representative Employee.

#	Acronym	Application Name
1	DIR	Directory
2	HRS	Human Resources
3	FIN	Financial Application
4	CLA	Administrative Application
5	CLB	Collaboration
6	ISA	Administrative Application
7	PTL	Staff Application

Table 3.3: Application Roles Required by our Representative Employee.

Acronym	# of Roles
DIR	2
HRS	2
FIN	2
CLA	1
CLB	2
ISA	3
PTL	1
	13

more challenging. Capturing application requirements by staffing position, as detailed in Table 3.2, facilitates the on-boarding process in enterprise organizations by formalizing access control requirements in an intuitive, scalable and dynamic framework that may be automated or captured in formalized business processes. Consider the impact when Alice retires, for instance. If the relationships constructed in both Table 3.2 and Table 3.3 remain when a new employee resources the Administration Officer staffing position, this could greatly facilitate the on-boarding process.

In summary, this case study shows that for one employee of the GoC, a total of 7 applications are required to fulfill the responsibilities of the staffing position. In addition, a total of 13 roles are indirectly associated with the staffing position. While we understand that many of these roles are meant to be shared by multiple employees, we also recognize that we have real-world examples for both Symptom 1 and 2 of role explosion as defined above. We see that an organization requires employees to access several applications autonomously managing their own set of role information. We also see that an organization has one or more applications where the ratio of roles to subjects approaches or exceeds 1 (Table 3.1).

In this dissertation, we refuse to dismiss the 150% role to subject ratio discovered in our case study as bad RBAC design. Instead, we are curious why this occurs. In the discussion that follows, we make simple observations related to influential RBAC literature found in the previous chapter.

3.5 Discussion

In 2001, Schaad, Moffett and Jacob perform an empirical study for the role-based access control system of Dresdner, a large European bank, highlighting the fact that *very little has been done to identify and describe existing role-based access control systems within large organizations* [76]. In their analysis of the number of roles for this real-world example they theorize that the total number of roles would be the product of every position and function in the organization. They identify 65 official positions, ranging from Clerk to Branch Manager and 368 different job functions, as provided by the Human Resources Management System (HRMS), suggesting a possible 23,920 roles. However, the number of implemented roles they discovered *currently in use* was about 1300 for 40,000 subjects, a ratio of 3-4% matching the estimate provided at the RBAC2000 Workshop.

Schaad et. al elaborate on these findings, identifying that this distribution is not uniform due to the pyramid shaped hierarchy of many organizations, like Dresdner Bank, where there are always many more clerks than there are Heads of Divisions. It is important for the reader to pause and reflect on the relevance of these findings nearly two decades later, especially considering that the access control system under study, called FUB, was already more than a decade old and suffering from several limitations and weaknesses including the following:

- A user can be assigned to more than one role, for instance when a colleague becomes ill or is on holidays, but also in more permanent cases where a clerk works at one branch in the morning and another in the afternoon. In the RBAC96 model a user (or subject) must choose which role to activate. However, with FUB there is no concept of a session, so when a user logs into a system they have all the permissions of all the roles they have been assigned. This creates a problem with respect to the principle of least privilege and careful attention must be paid by administrators in these scenarios.

- Access control services are provided to users not considered permanent staff, including consultants and temporary employees who work for the Bank during projects of varying length. Due to administrative overhead this group is not included in the HRMS and must be manually administered by FUB staff. The resulting overhead is substantial as there are hundreds of users in this group. Furthermore, these numbers are not included in the role to subject ratio presented above.
- FUB only permits administrators to group employees using the combination of function, position and organizational unit. It is not possible to group employees according to other criteria and assign group-specific access rights.
- The implementation does not allow for the grouping of access rights which naturally belong together. For example, it is not possible to group the privileges required to *create account* and *delete account*. Instead, these access rights must both be granted to each and every role requiring these permissions. It would be better if these privileges could be grouped to facilitate administration.

Only five years later, in 2006, mounting evidence for the *flattening* of corporate hierarchies was being produced [67]. Rajan and Wulf examine how corporate hierarchies have changed in the recent past using a detailed database of job descriptions and their reporting relationships tracked over a period of 13 years. First, they discovered that the number of managers reporting to the Chief Executive Officer (CEO) has increased steadily over time from an average of 4.4 in 1986 to 8.2 in 1998. Second, they find that the depth or number of positions between the CEO and the lowest managers has decreased by more than 25% over this period. Finally, they provide three explanations as follows:

- An increase in the competitiveness of the external environment forcing the need for a more streamlined organization
- An improvement in corporate governance, forcing CEOs to eliminate excessive layers of managers built up during past empire building
- Advances in information technology that expand the effective span of control for top managers

We observe scalability concerns for each administrative RBAC model described in the previous chapter, beginning with the ARBAC family of models [74][71][63].

ARBAC97 describes the decentralized administration of subject-role enrollment, role-role grants and permission-role assignment with reference to the RBAC96 model [72]. In their introduction to ARBAC97 the authors presume that *in large enterprise-wide systems, the number of roles can be in the hundreds or thousands, and subjects in the tens of hundreds or thousands* suggesting that the ratio of roles to subjects is 10%. The belief, notion or sense that the number of subjects far exceeds the roles found in enterprise systems is repeated in each extension to the ARBAC97 model as is the example of a Director overseeing two projects with a Project Lead, Production Engineer, Quality Engineer and a (Junior) Engineer. However, if one considers the simple example of one employee in each role on each project we observe that this example defines eleven roles for nine subjects, a ratio of more than 120%. Furthermore, when we conceptually scale this model up to hundreds of projects in a society where the demand for skilled workers is not being met [41], we wonder whether Junior Engineers are allocated in quantity to single or multiple projects.

SARBAC is intended to be used with RBAC96 as a complete role-based model for administration [16]. Unlike ARBAC, SARBAC does not assume the existence of a disjoint set of administrative roles. SARBAC develops a model for role hierarchy administration with the belief that it will be easier to then incorporate subject-role and permission-role administration. We observe that Crampton et al. reuse the example from ARBAC97 where the ratio of roles to subjects is more than 120%.

A-ERBAC describes the model employed in a commercial enterprise security management software solution [42]. Kern et al. suggest that Enterprise Roles are increasingly used by medium to large organizations as the basis for security management across different systems. A-ERBAC uses the concept of *scopes* to control the authority of administrators on a Target System. An administrator may be assigned the ability to view, insert, change or delete various RBAC elements such as subjects and roles provided they are assigned one or more administrative scopes within the hierarchy of objects (e.g. Organizational Units or Cost Centres). Furthermore, Kern et al. argue that the scopes of A-ERBAC provide a more comprehensive solution than the *pools* of ARBAC02 as each scope is optionally associated with attributes that enrich the administrative convenience. This work also provides a case study from a European bank where subjects are created and deleted using connections to the Human Resources database.

It is not explicitly stated in this work that Dresdner bank is the institution observed [76]. If indeed this case is based on another banking institution with 70,0000 subjects (vice 40,000 at Dresdner) it would have been interesting to know the number of roles implemented in this system. Instead, we observe that the example with functional and business roles depicted in this work identifies a scenario where eight roles are defined for five subjects. We understand that this disproportionate use of subjects and roles is not the focus of the A-ERBAC model and acknowledge that the authors assume the reader will intuit that as this example scales more subjects will be disproportionately enrolled in the defined business roles. Nevertheless, we find it interesting that the ratio of roles to subjects is 160% in the example depicted.

With all due respect to the Dresdner bank case study, consider the classic RBAC example of the *bank teller* role where hundreds of subjects share a generic, simplistic role with the exact same permissions. We argue that this scenario is still well-ingrained in the collective psyche of the research community and we have found it difficult to challenge this long held belief and gain traction. Anecdotally, the rise of on-line banking, automated teller machines and the proliferation of credit card transactions has impacted the number of employees, specifically bank tellers employed at banks. It would be interesting to review the ratio of subjects to roles at Dresdner bank today. We suspect that if one was to monitor this ratio on a yearly basis there would be a clear trend.

In the previous subsection, we offer a new case study where the number of roles exceed subjects. This may seem counterintuitive to those visualizing a classic RBAC example where tens or hundreds of subjects are enrolled in the *bank teller* role which has been assigned several permissions.

It is not clear whether the classic subject-role-permission model remains dominant in medium to large organizations with highly skilled workforces. Over a decade and a half ago, the A-ERBAC model described a second layer of roles between subjects and permissions without expressly highlighting the fact that this implies subjects will always hold a minimum of two roles in a hierarchy (i.e. subject-role-role-permission). We understand that these roles are meant to be shared but we believe there has been a fundamental shift in the way RBAC systems are implemented and maintained over the past decade.

In our investigation, we are unable to determine if the research community influenced the introduction of subject-role-role-permission design patterns or whether this was an organic by-product of real-world implementations or both. In 2000, Oh and Park describe the notion of Task-Role Based Access Control (T-RBAC) where intermediary tasks are assigned permissions instead of roles [62]. In 2001, Schaad, Moffett and Jacob describe roles hierarchies based on positions and functions [76]. Peisert and Bishop suggest that traditional access controls have evolved from being static and coarse-grained to being dynamic and very fine-grained [66].

3.6 Summary

In this chapter, we describe *role explosion*, conducting a real-world case study for one employee in the GoC to better appreciate this concern and further motivate a return to first principles for RBAC. We define two symptoms for role explosion and perform an empirical analysis for one GoC organization, discovering the following real-world results:

- At the macro level, we discover a role to subject ratio exceeding 1. As detailed in Table 3.1 the role granularity metric for one application exceeds 150% in 2008.
- At the micro level, we perform an empirical analysis for one employee of the GoC requiring access to several applications autonomously managing their own set of role information as listed in Table 3.3.

In this chapter, we satisfy our first objective, challenging the long held belief, notion or sense that the number of subjects far exceeds the roles found in enterprise systems. If role explosion is a normal occurrence and previous models for the administration of RBAC consider role explosion a design problem, this should motivate the requirement for new RBAC structural relationships.

The reader should now better appreciate why role explosion occurs in large organizations. It is rarely a dramatic explosive event but rather an organic daily growth pattern for large organizations where *more and more new types of applications that require controlled sharing of resources or discrimination of information appear* [9].

In this dissertation we argue in favour of role evolution, introducing our new ORGODEX model and methodology to direct and control growth. We acknowledge role explosion but do not consider it a design problem. Instead we motivate the demand for our a new evolutionary growth pattern for large organizations whose employees require access to information on a need to know basis. In the following chapter, we compare the information security of ACME university to RBÄC, using real-world data and experiences to support this new principled approach to RBAC, or role evolution. In chapter 5 we introduce our new ORGODEX model and methodology for engineering scalable access control systems and distinguishing between roles and responsibilities.

3.7 Publications

Publications related to this chapter:

- Aaron Elliott and Scott Knight. Role Explosion: Acknowledging the Problem. In Proceedings of the 2010 International Conference on Software Engineering Research and Practice, WORLDCOMP, pages 349-355, 2010. CSREA Press.

- Aaron Elliott and Scott Knight. One Employee and Several Applications: An Information Management Case Study. In Software Engineering Research and Practice, WORLDCOMP, pages 179-185, 2009. CSREA Press.

4 Role Evolution

4.1 Introduction

In the previous chapter we discussed role explosion, performing a case study to better appreciate this concern and its origin. In this chapter, motivated by role explosion, we discuss the notion of directed evolutionary growth for RBAC implementations. This dissertation is divided into three distinct phases:

- In this chapter, we further refine our contributions, introducing and motivating the requirement for new RBAC structural relationships that distinguish between roles and responsibilities, thereby satisfying our first objective. We analyze a real-world RBAC implementation, performing an empirical study at ACME university.
- In chapter 5, we introduce our new ORGODEX model and methodology for engineering scalable access control systems, thereby satisfying our second and third objectives.
- In chapter 6, we validate the efficiency and practicability of our model and methodology during a real-world empirical study, leading an initiative to deploy an instance of ORGODEX at two geographically distributed partner institutions.

In the following sections, we refine our contributions. First, we analyze a real-world access control system, performing an empirical study at ACME university and compare its RBAC implementation with RBÄC, first introduced in chapter 2. We use real-world data and experiences in support of this new principled approach to RBAC, further extending RBÄC with role-centric constraints and describing how ACME university decouples subject and permission management, using role evolution to direct growth.

Next, we further motivate the requirement for new RBAC structural relationships to describe and reason about implementations, We advocate on behalf of role evolution, a mandatory divergence between roles and responsibilities. Then, we investigate how ACME university performs access man-

Table 4.1: ACME Roles as at April 2015

Label	**Count**
Subject	351
Role	558

agement, performing a microanalysis on a scenario of special interest, using our new hierarchical graphing model to better visualize the subject-permission mappings and introducing our new role-centric methodology for dynamically constraining access to information [20].

Finally, we discuss why RBÄC provides a stabilizing layer for RBAC and we describe a fundamental shift in the way RBAC implementations are engineered and maintained. At the conclusion of this chapter, the reader will better understand the requirement for role evolution, a semantic divergence between roles and responsibilities when engineering and maintaining RBAC implementations.

4.2 Background

ACME university is located in North America. While the name of the institution is contrived for anonymity, the information that follows is based upon an existing RBAC implementation, supporting more than 700 employees. We perform queries against the data dictionary to determine the complete set of roles created in the IM system[1]. From this set we have excluded subjects and roles created at the installation of the database software[2].

As listed in Table 4.1 we discover that ACME university has a role to subject ratio of approximately 160%. Coincidentally, this is the ratio we describe in section 3.5 for ARBAC97, SARBAC and A-ERBAC. This is clearly a huge deviation from the 3-4% ratio estimated at the RBAC2000 Workshop or the 10% ratio described by Sandhu et al. in ARBAC97. Upon further investigation of the available metadata, we learn that the ACME university role information is well-documented within the IM system, greatly facilitating this research.

[1]Relational databases typically include meta data repositories identifying objects created in the system such as roles

[2]Oracle® databases include several roles such as *DBA* that were considered out of scope for this study

Table 4.2: ACME Grants as at April 2015

Grant	**RBÄC**	**Count**
Subject-Role	SR^+	386
Role-Role	RH^+	294
Role-Role	G	683
Role-Role	DH^+	215
Permission-Role	PD^+	2281

Each and every role is labeled an Appointment, a Position, a Group or a Functional role in a table called *Role Documentation*. Understanding that RBÄC proper roles (R+) are granted to subjects and do not obtain permissions directly, we determine that Appointment, Position and Group roles meet this description. Like RBÄC demarcations (D+), Functional roles are assigned permissions directly and granted to proper roles.

Subjects are enrolled into Appointment roles with no direct permissions as described in RBÄC (SR^+). For example, when a faculty member is appointed as Departmental Chair #1 for a three year term, the HR group enters this information into the IM system. This data entry is used to automate the enrollment of a subject into the corresponding *Department Chair #1* role for a three year term.

Subjects were also enrolled into Position roles on an indeterminate or term basis (SR^+). Employees hired indeterminately are identified by a job title or position number and enrolled in the corresponding role. Unlike Appointment roles, the enrollment of subjects into Position roles is not automated. We understood this automation would be introduced at a future date.

With consideration for the examples presented in ARBAC, SARBAC and A-ERBAC this seems reasonable. ARBAC and SARBAC describe a model that is based on the positions found in the Engineering Department of a fictitious organization. A-ERBAC uses the example of business roles, similar to positions, in their account of the design found in a European bank.

Using the *Role Documentation* and the data dictionary allowed us to determine the subject-role, role-role and permission-role mappings. Our query results are listed in Table 4.2. As further evidence that the RBAC implementation under investigation is indeed a practical example of RBÄC we queried role-role grants. We observe that Appointment and Position roles are often enrolled into Group roles forming proper role hierarchies (RH^+). As one example, the role *Departmental Chair #1* is enrolled into the *Departmental*

Chair Group role. Similarly, we observe that Functional roles are often assigned to one another in demarcation hierarchies (DH$^+$). As one example, the role *Approve Grades* is assigned to the role *Final Grades.* Finally, we observe that Functional roles are granted to Appointment, Position and Group roles consistent with the Grant (G) relation of RBÄC.

4.3 Divergence

In section 2.10, we review RBÄC, a new fragment of RBAC that may be applied to both new and existing implementations [46]. Unlike previous extensions to the classic RBAC model,RBÄC revisits first principles, extending the traditional subject-role-permission, or *triangular model*, with a second mandatory role to form the subject-role-role-permission *square model* (Figure 2.6).

Not only does RBÄC improve organizational scalability, it improves comprehension by separating proper roles (i.e. roles) from demarcations (i.e. responsibilities). Although communication with the business side of an organization does not appear to be a primary motivation for RBÄC, we suggest the formalization of this additional layer is an important evolutionary concept, enforcing a divergence between roles and responsibilities for RBAC.

A dichotomy partitions its membership into two distinct subsets where everything must decidedly belong to one set or the other. In this section, we motivate the requirement for a new RBAC dichotomy, partitioning roles and responsibilities into distinct subsets. In large RBAC implementations with hundreds or thousands of roles, the phylogeny of roles is an important evolutionary concept. It is no longer sufficient to think simply in terms of roles. In Figure 4.1, we present a new phylogenetic representation for RBAC systems where responsibilities have semantically diverged from roles. This is an important evolutionary concept because there is considerable misunderstanding among information security practitioners, where access control is often performed irregularly and considered a secondary duty [9].

Definition 18. A **Role** is a part that someone or something has in a particular activity or situation. For example, a role or position in an organization like a Departmental Chair (DC).

Definition 19. A **Responsibility** is a duty or task that one is required or expected to do. A DC is responsible for approving grades.

Using the terminology role and responsibility is a practical approach to implementing RBAC, permitting engineers to more easily communicate the im-

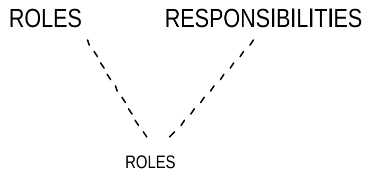

Figure 4.1: Role Evolution proposes a phylogenetic divergence between Roles and Responsibilities.

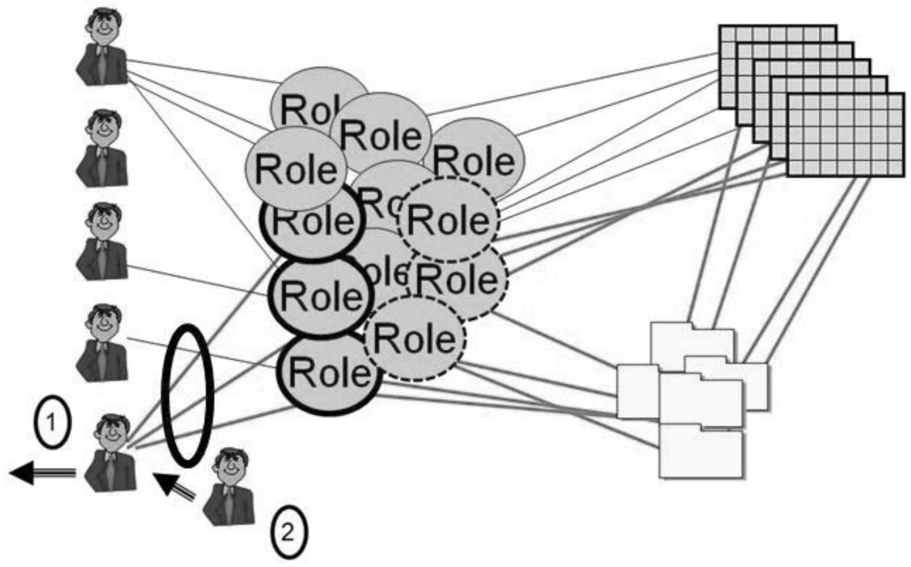

Figure 4.2: Subject-role relationships may be lost due to employee turnover.

portant aspects of their design implementations to each other and the business side of an organization. This is an important concern because role engineering is the most expensive aspect of deploying RBAC [58].

Using organizational structure as the security framework provides clear focus for business analysts charged with eliciting requirements. Unlike Feltus et al. [25], we do not believe responsibilities should ever be directly assigned to subjects. Our argument against *direct atomic responsibility assignment* relates to employee turnover.

In large organizations with highly skilled workforces, an employee (i.e. subject) may acquire several responsibilities directly, resulting in the loss of important subject-role relationships when an employee leaves the organization. In Figure 4.2, we observe that if employee #1 leaves the organization three subject-role relationships may be lost before they are assigned to employee #2. Like the authors of RBÄC, we believe a *conceptual split right down the middle* of RBAC improves scalability, eradicating inflexible subject-role-permission *triangles* from large organizations. Furthermore, we have observed how this conceptual split facilitates information management security in practice, insulating systems against ongoing employee turnover, policy changes and reorganization.

It is not clear whether the classic subject-role-permission model remains dominant in medium to large organizations with highly skilled workforces. Over a decade and a half ago, the A-ERBAC model described a second layer of roles between subjects and permissions without expressly highlighting the fact that this implies subjects will always hold a minimum of two roles (i.e. subject-role-role-permission). We understand that these roles are meant to be shared but we believe there has been a fundamental shift in the way RBAC implementations are maintained over the past decade.

Conway's law suggests that managers should take a step back from looking at a system with respect to inputs, outputs, and call graphs and, instead, examine the tasks that people must perform and how software modules influence these tasks [47]. Coyne suggests that the definition of roles is essentially a requirements engineering process [13]. Unlike traditional top-down or bottom-up approaches for role engineering [55], RBÄC proposes an additional layer of abstraction such that permissions are never assigned to subjects in the traditional triangular subject-role-permission model. Instead, permissions are **always** assigned indirectly using a square subject-role-role-permission model, thereby facilitating organizational scalability. Consequently, the problem of determining the optimal set of roles for an RBAC implementation, referred to as the Role Mining Problem (RMP) is fundamentally redefined in the context of a square model, where finding the minimal set of roles (e.g. Basic-RMP) is not a primary concern [83].

In the following section we extend RBÄC, adding role-centric constraints in support of directed evolutionary growth for RBAC implementations. Our extension acknowledges and supports a common reality for large organizations where responsibilities may be shared by various members of the workforce, but the scope is different. For example, at ACME university, the group DC shares similar responsibilities but their accountability relates to different sets of degree programs, courses and students. We use our new hierarchical graphing model to better visualize the subject-permission mappings to depict how our constraints permit shared responsibilities to be scoped.

4.4 Constraints

In the previous section we describe a divergence between roles and responsibilities. Like the authors of RBÄC, we advocate on behalf of role evolution, using the terms role and proper (role) interchangeably throughout this section. We also use the terms demarcation and responsibility synonymously, acknowledging that our work is rooted in this semantic divergence for RBAC implementations. In this section we describe how ACME university performs access management, describing the relationship between the role *Departmental Chair #1* and the responsibility *Approve Grades.*

Example 1. Departmental Chair #1 is responsible for Approving Grades.

- s_1 = Dr. George Scott
- p_1 = SELECT information FROM course_view
- R^+ = {Departmental Chair #1, Departmental Chair}
- RH^+ = {(Departmental Chair #1, Departmental Chair)}
- D^+ = {View Final Grades, Approve Grades}
- DH^+ = {(View Final Grades, Approve Grades)}
- SR^+ = {(s_1, Departmental Chair #1)}
- PD^+ = {(p_1, Approve Grades)}
- G = {(Departmental Chair, View Final Grades)}

At the end of each academic term, Departmental Chair #1 is responsible for approving final grades. Instructors set up an evaluation scheme and evaluate the student but before the final grades are released they must be approved by a Departmental Chair. The implicit SP mapping required for DC #1 to approve final grades is explicitly *spelled out* in the language defined by RBÄC. The subject is Dr. George Scott (s_1) and the permission is SELECT information from course_view (p_1). In other words, Dr. George Scott, as DC #1, must have access to a constrained list of courses from the IM system so that he may review and approve grades.

In Figure 4.3 we visualize our scenario using the graphing model introduced by Kuijper and Ermolaev [46]. On the left side of the graph are the proper roles DC #1 and DC. On the right side of the graph are the demarcations View Final Grades and Approve Grades. In this example, we have the implicit SP relationship and the explicit RBAC implementation where

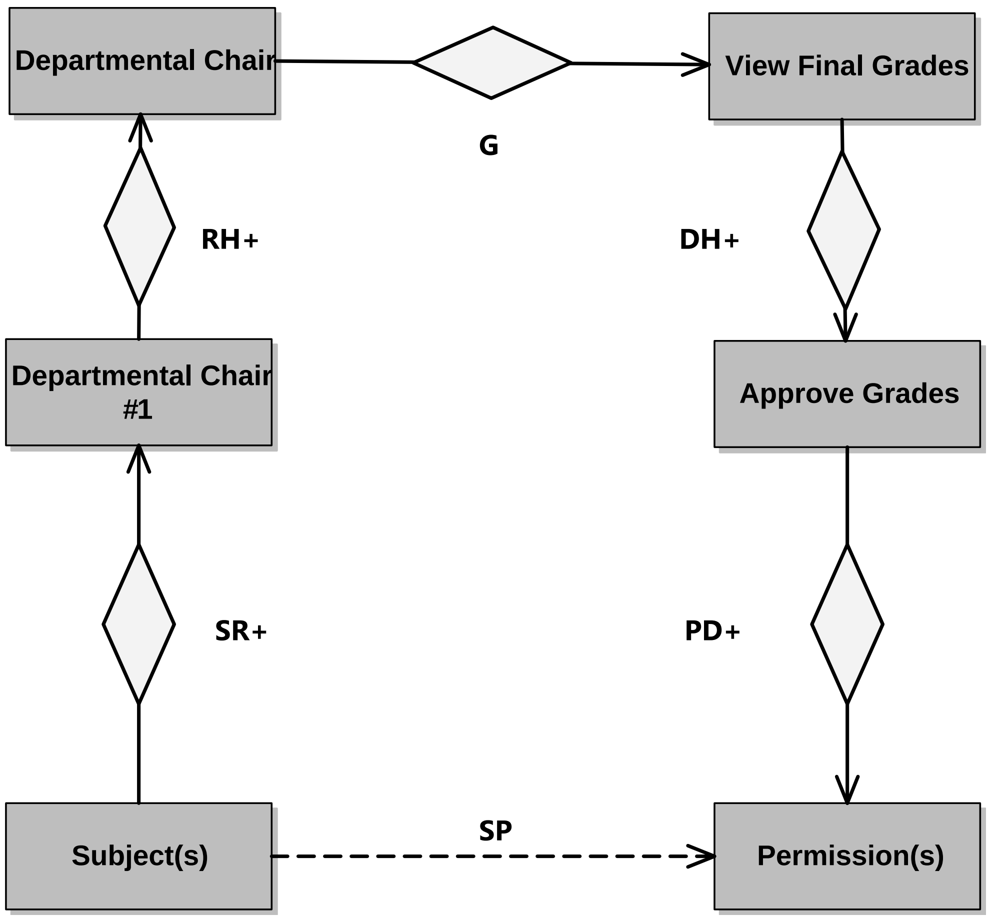

Figure 4.3: An RBÄC example found at ACME

subject-role-role-responsibility-responsibility-permission relationships permit DC #1 to approve grades. SR^+ indicates a subject-role relationship, RH^+ defines a role hierarchy, G indicates that a responsibility (or demarcation) is granted to a role, DH^+ identifies a demarcation (or responsibility) hierarchy and PD^+ describes a permission-demarcation (or permission-responsibility) relationship.

At this juncture, the reader may be curious why ACME university has an RBAC implementation where five relationships exist between the subject Dr. George Scott and the permission required to approve final grades. In Figure 4.4 we introduce our new hierarchical diagramming notation to better visualize the directional hierarchy of subject-role, role-role and permission-role

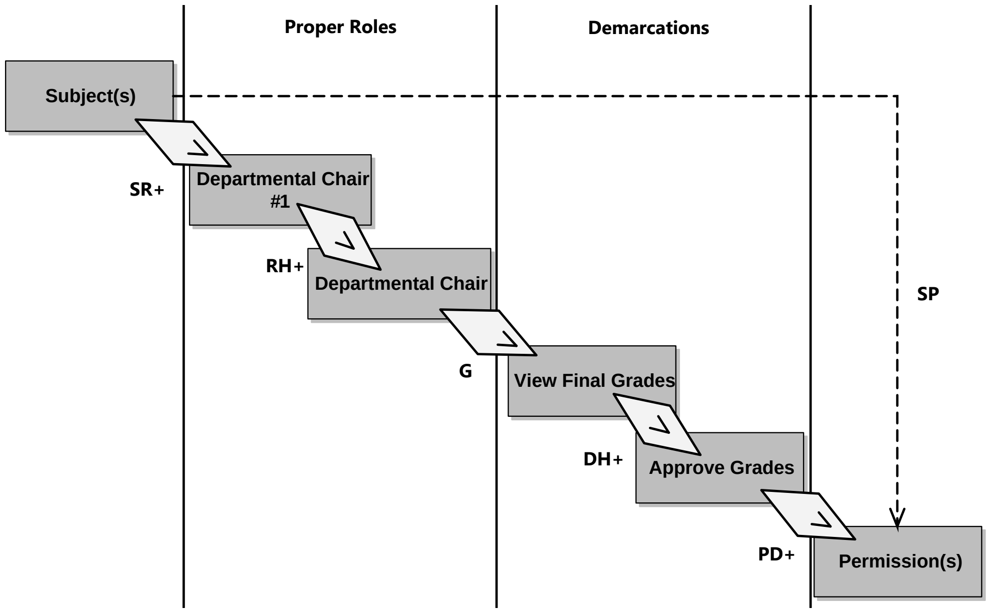

Figure 4.4: Hierarchical Graph for the example found at ACME

grants, explaining the rationale for this design.

The elements of our diagram can be reconfigured to look like the *desirable square (or rectangle)* of RBÄC however, we feel that the notion of hierarchy is an important aspect not well represented in Figure 4.3. Instead, we use *swim lanes* to depict the conceptual boundaries between subjects, proper roles, demarcations and permissions. To better comprehend real-world RBAC implementations it is important to visualize the directionality of the enroll, grant and assign relationships and *see* the depth at which one actually obtains permissions to do something. In our scenario of interest there is a cascade of diamond shaped relationships between entities (i.e. rectangles) ultimately linking subjects and permissions together. We use arrows within the diamonds to indicate directionality. When one analyzes Figure 4.3 and Figure 4.4, it is important to understand the implicit relationship SP is non-trivial. There are five relationships explicitly defined between Dr. George Scott and the permission SELECT information from course_view. As we will see, this new hierarchical diagramming technique also facilitates the introduction of our new role-centric constraints.

In this section we present a new methodology for dynamically constraining permissions under RBÄC, providing additional validity for this new fragment of RBAC. We elaborate upon our scenario of interest where DC #1 is responsible for approving final grades at the end of each academic term. Unlike ARBAC and SARBAC which describe constraints on subject-role enrollments and permission-role assignments our constraint model introduces constraints on permissions already assigned to one or more roles. Unlike RABAC [39], whose intent is to dynamically constrain the set of permissions available to users, our intent is to dynamically constrain the information returned by static role-responsibility-permission relationships.

At ACME university, each DC is responsible for approving final grades within their respective department. We observe that each DC is directly enrolled in a proper role indicative of their appointment, meaning one unique role for each and every Departmental Chair. This is counterintuitive. Classic RBAC suggests that all subjects share a *grouping* role called *Departmental Chair* in a triangular subject-role-permission design. However, we discovered that instead of subjects sharing a grouping role, there were eleven unique *Departmental Chair* roles (i.e. #1 .. #11) enrolled in a *grouping* Departmental Chair role for a total of twelve defined roles. Using Figure 4.5 as a point of reference, consider the result. One could draw eleven diagrams exactly alike, simply substituting the *Departmental Chair #1* role with each of the other Departmental Chair # roles.

We introduce constraints for RBÄC as a (name,value) pair attribute restricting role-responsibility grants. For example, DC #1 is responsible for Approving Grades in their respective department and may only SELECT courses WHERE the attribute DEPT is restricted to the value [#1] when Reviewing Course Information (Figure 4.5).

Example 2. DC #2 is responsible for Approving Grades in their respective department. DC #2 may only SELECT courses WHERE the DEPT=[#2] when Reviewing Course Information (Figure 4.6)

This was a recurring theme. We learned that DC #1 had SELECT access on similarly restricted lists of students, programs and staff within their respective department. In each case, the WHERE clause used a role attribute to restrict access to information. Figure 4.6 depicts the *match* condition used to constrain the List of Courses for both DC #1 and DC #2 when approving final grades.

We raised our concern with the practice of assigning the Department (name,value) pair to the applicable role vice simply using the Department assigned to the person in the HRMS [34]. In response to our concern, we were informed that the department of a subject or individual did not always reflect an appointment. We were informed that Deans, for instance, were faculty

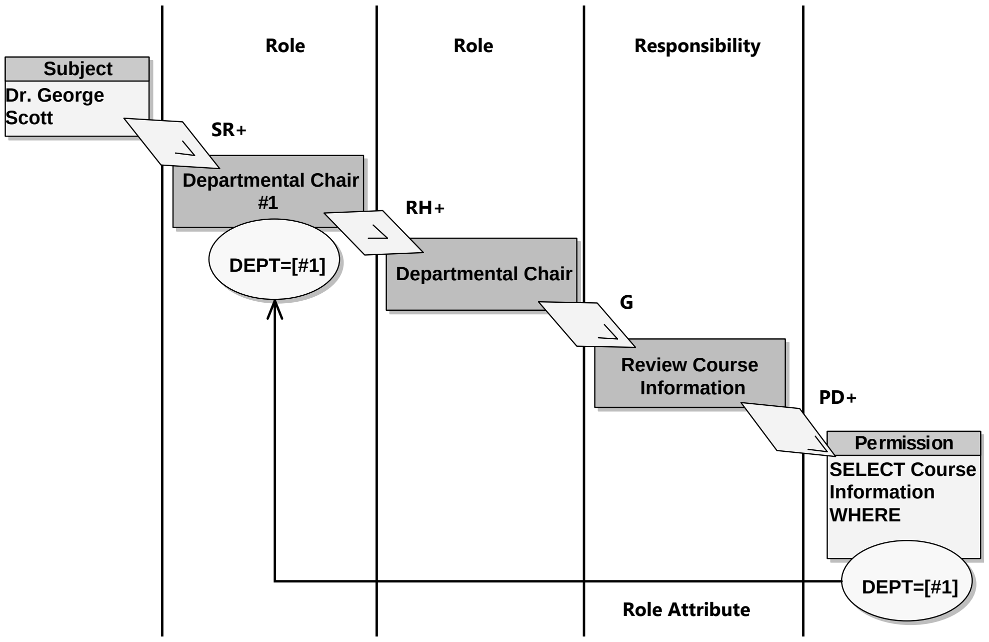

Figure 4.5: Constraining RBÄC. The List of Courses that may be selected by Departmental Chair #1

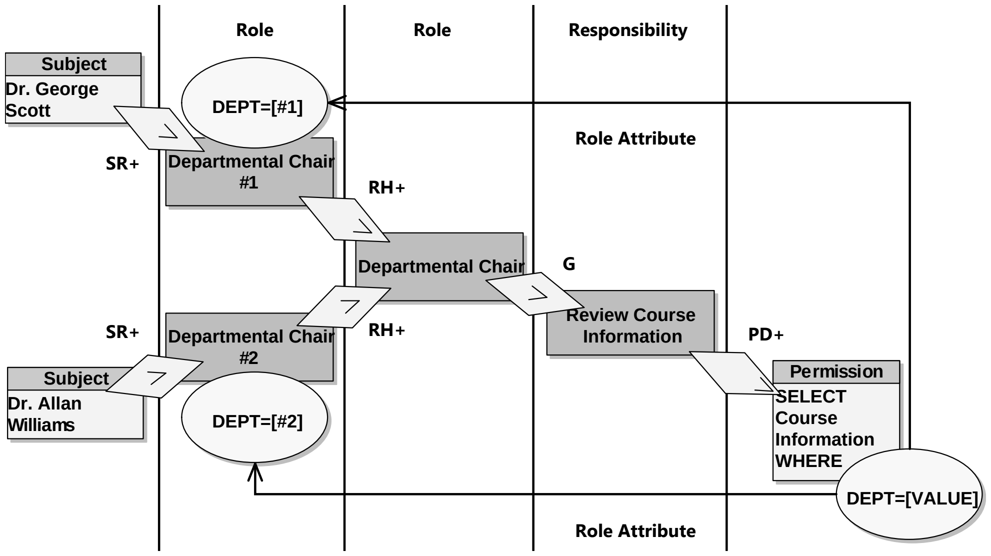

Figure 4.6: Determining the List of Courses that may be selected by two Departmental Chairs.

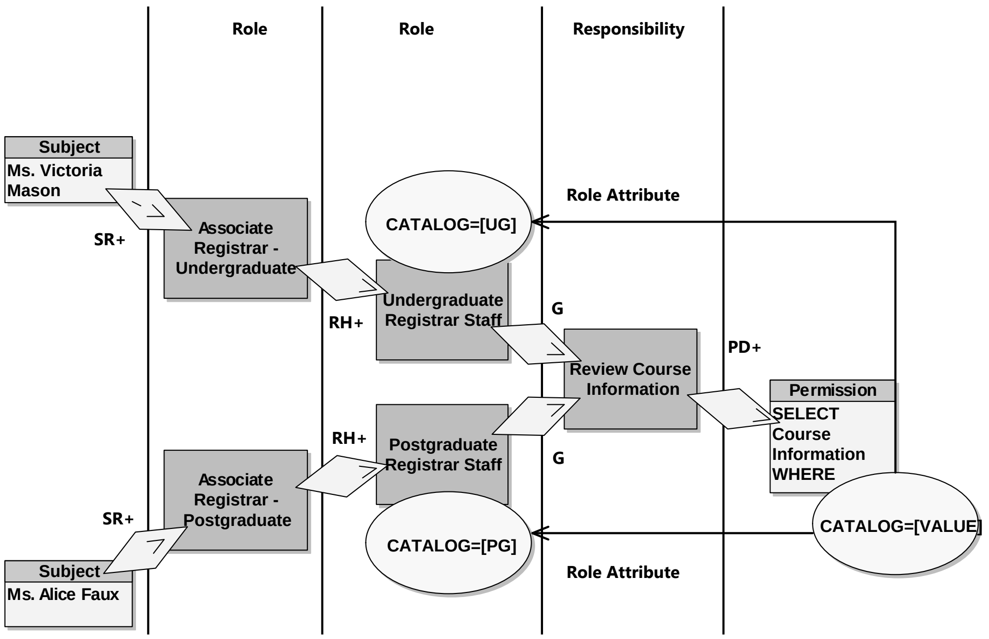
Role
Role
Responsibility
Subject
Ms. Victoria Mason
SR+
Associate Registrar - Undergraduate
CATALOG=[UG]
Role Attribute
Undergraduate Registrar Staff
RH+
G
Review Course Information
PD+
Permission
SELECT Course Information WHERE
RH+
Postgraduate Registrar Staff
G
Associate Registrar - Postgraduate
SR+
CATALOG=[PG]
CATALOG=[VALUE]
Role Attribute
Subject
Ms. Alice Faux

Figure 4.7: Determining the List of Courses that may be selected by two Staff Members.

members within one academic department while simultaneously responsible for an entire faculty under their appointment.

Figure 4.7 depicts the *match* condition used to restrict the list of courses for both the Associate Registrar, Undergraduate (AR-UG) and Associate Registrar, Postgraduate (AR-PG). In this instance, we see that the role attribute, or (name, value) pair (Catalog, [Value]), is used to restrict access. When the AR-UG accesses course information the name value pair (Catalog, [UG]) is used in the WHERE clause, and when the AR-PG accesses course information the name value pair (Catalog, [PG]) is used in the WHERE clause.

Example 3. The AR-UG is responsible for maintaining UG Course Information. The AR-UG may only SELECT courses WHERE the CATALOG=[UG] (Figure 4.7)

Example 4. The AR-PG is responsible for maintaining PG Course Information. The AR-PG may only SELECT courses WHERE the CATALOG=[PG] (Figure 4.7)

For ACME university, we were inundated with examples of the practicality of this approach, especially when considering the degrees of freedom afforded to application architects. We were informed that the relative cost of creating and maintaining database objects such as views and packages was relatively expensive. On the other hand, the cost of creating and maintaining light-weight roles that could be easily delegated with no direct permissions was inexpensive. For this reason, the Security Architect had decided to *hide* this design decision within the role information, affording applications more flexibility and scalability. Organically, from the year 2000 onward, ACME had discovered that who has access to what was difficult to maintain in an environment with constant employee turnover. By aggregating permissions into *responsibilities* (i.e. demarcations) and assigning them to Appointment, Position or Group roles (i.e. proper roles) in a role hierarchy they could avoid losing important security relationships when employees left the organization (Figure 4.2). If a subject had acquired several permissions directly or indirectly via grouping roles this was previously lost when subjects were removed from an RBAC implementation. To avoid this loss, permissions are aggregated into *responsibilities* and assigned to Appointment, Position or Group roles.

4.5 Discussion

RBÄC provides a stabilizing layer for RBAC. It facilitates the automation of enrolling employees into proper roles (SR^+) and it prevents the loss of important security relationships when employees leave the organization. In this chapter we introduce dynamic role-centric constraints for RBÄC as a means of enforcing conditionals when subjects share the same responsibility in different departments, for instance. This is an important design concern for medium to large organizations operating under a *need to know* security policy.

Role evolution presumes the number of roles will scale uncontrollably in large organizations or controllably using a prescribed methodology. We acknowledge that *responsibilities explosion* may be a concern for role evolution. However, we believe the optimization work found in the role engineering literature could be leveraged to refine the number of responsibilities defined by an RBAC implementation [55]. In particular, bottom up role mining might be an interesting means of addressing this concern [83].

Responsibilities explosion is a natural occurrence emerging organically over the lifetime of an RBAC implementation, as measured in months and years. A simple example would be a poorly named responsibility that is duplicated when a Security Architect does not realize a suitable responsibility for their requirement already exists. With role mining, responsibility duplication could be reported, addressed and controlled. In addition, role evolution controls explosion through the use of role-centric constraints, permitting the reuse of responsibilities by different roles.

Role evolution defines an architecture where roles are not explicitly assigned permissions. Orphaned roles no longer needed by an organization may be dropped inconsequentially. In contrast, orphaned responsibilities, permitting access to information must be more carefully considered and only removed on confirmation from the business side of an organization. With the controls in place to *semantically* check and act upon the population of roles and responsibilities, entitlement review might equate to addressing *alerts* raised by an application instead of monthly or bi-annually attempting to review reports where business terminology is not being used.

With role evolution, a primary concern is communication with the business side of an organization. We are motivated to better align role and requirements engineering with a methodology that facilitates communication, using roles and responsibilities to describe and reason about RBAC implementations using business terminology. We recognize that there is a mismatch between mental models and real-world RBAC implementation where designs and operations are often trivialized when individuals carry invalid assumptions from the physical world into the digital world [54].

We acknowledge that it is non-trivial and irregular to think of responsibilities absent those charged with their completion. Nevertheless, we contend that responsibility analysis is an essential component of role evolution. We suggest this based upon decades of practical experience as a Security Architect in an organization where responsibilities dynamically shift on a weekly, monthly and yearly basis. For RBAC implementations, we contend that this is a fundamental change, a new paradigm extending the scalability and maintainability of the RBAC model.

Similar to *Role Templates* as proposed by Giuri [29], we recognize the requirement for permission granularity in Relational Database Management Systems (RDBMS) and our constraints model addresses this need. For example, granting select permissions on a database table to a role enables all subjects with this role to access every record in the table unless constraints are put in place. In this chapter we do exactly that, however unlike Role Templates, our RBAC extension permits constraints for role-responsibility relationships. It does not require administrators to constrain access for individual subjects, nor does it require database views to be constructed for each role. This is important because it minimizes the maintenance effort in large organizations where employee turnover and security policy changes render the creation of database views for each *parameterized role* and direct permission grants for individual subjects impractical [7].

As with Extended RBAC [85], our constraints model does not permit attributes to be associated directly to subjects. For the reasons described above this is impractical in large organizations where employee turnover and term appointments, like DC #1, necessitate role-attribute relationships to promote an economy of mechanism [51]. However, unlike the Extended RBAC model our constraints do not associate attributes with roles directly, instead we place constraints on the relationship between a role and a responsibility. This is important because the granularity of responsibilities dictate that a DC have more or less permissions depending upon the task at hand. Specifically, a DC might be responsible for reviewing course information for an entire department while simultaneously restricted to entering grades for the courses they are instructing.

The insider threat has long been considered one of the most serious challenges in computer security [10]. Unlike external threats, insiders require access to information to perform their job, creating a *gap* between the practicality of administering one's access control system and the principle of least privilege, thereby exposing too much information and creating susceptibilities for abuse. Role evolution seeks to bridge this gap with a semantic divergence between roles and responsibilities, resulting in a clearer mental model for RBAC implementations and facilitating entitlements review [54].

In this chapter we describe how organizational scalability is enhanced at ACME university by decoupling subject and permission management at the expense of role evolution. During our analysis at ACME university, we observed that the Security Architect had aggregated permissions into responsibilities and assigned them to Appointment, Position or Group roles in a role hierarchy to avoid losing important security relationships when employees left the organization (Figure 4.2). In large organizations with highly skilled workforces, an employee (i.e. subject) may acquire several responsibilities directly, resulting in the loss of important subject-role relationships when an employee leaves the organization. Furthermore, we have observed how this conceptual split facilitates requirements engineering for information security. Like RBÄC, role evolution permits many-to-many administrative mutations and ultimately leads to more organizational scalability.

We observe that ACME university is a real-world instance of RBÄC where subjects and permissions are never linked by a single role. Instead, there is **always** *at least two roles* between a subject and a permission. We understand that this happened organically over the past fifteen years and was directly influenced by the ARBAC, SARBAC and A-ERBAC models [74][16][42]. Security practitioners at ACME university had discovered that who has access to what is difficult to maintain in an environment with constant employee turnover. By aggregating permissions into responsibilities (i.e. demarcations) and assigning them to Appointment, Position or Group roles (i.e. proper roles) they could avoid losing important security relationships when employees left the organization. This is exactly what RBÄC proposes with the following results:

- subject management is delegated as appropriate in organizations, reducing administrative overhead
- application architects are able to focus on creating independent responsibilities based on the functional requirements
- security officers are able to perform access management at an appropriate level of abstraction

In this chapter, our working example describes a scenario where eleven Departmental Chairs share similar responsibilities but their scope is different. At ACME university, role-responsibility constraints determine the records returned when reviewing course information. We learned that the role DC #1 was assigned the role-responsibility constraint, or (name, value) pair (Department, #1). Then when applicable queries were performed the WHERE clause used this constraint to determine what records should be returned. This was derived directly from the roles held by the subject. When Dr. George Scott, DC #1, reviews course information his role-responsibility constraints are used to determine whether or not courses may be viewed.

Finally, the fact that ACME university documents role and responsibility information is important to highlight. Not only did this practice greatly facilitate this research, it is directly responsible for our new concept of *role evolution*, the hypothesis being that if the business model of an organization directly informs its RBAC implementation the result is a security model that is more easily understood, more receptive to change and simpler to maintain.

4.6 Summary

In this chapter, we validate and extend the RBÄC model, making the following contributions:

- We introduce our new concept of *role evolution*, a divergence between roles and responsibilities
- We present our our new hierarchical graphing model to better visualize non-trivial SP mappings

- We extend RBÄC with our new role-centric approach for dynamically constraining access to information

Using real-world data and experiences, we compare the RBAC implementation of ACME university to RBÄC, detailing our scenario of interest for Departmental Chairs and using our new hierarchical graphing model to better visualize the subject-permission mappings before introducing our new role-centric methodology for dynamically constraining access to information.

The reader should now understand that role evolution is an extension of RBÄC, specifying a new mandatory layer for RBAC implementations where responsibilities are created, assigned permissions and granted to roles, thereby promoting organizational scalability. In the following chapter, we introduce our new ORGODEX model and methodology for engineering scalable access control systems.

4.7 Publications

Publications related to this chapter:

- Aaron Elliott and Scott Knight. Towards Managed Role Explosion. In Proceedings of the 2015 New Security Paradigms Workshop (NSPW), number 1, pages 100-111, 2015. ACM Press.

5 ORGODEX

5.1 Introduction

In the previous chapter, we suggest that the term role explosion be replaced by the notion of *role evolution*, presuming the number of roles will scale uncontrollably in large organizations or controllably using a prescribed methodology. In this chapter, we prescribe a new evolutionary model for RBAC in support of scalable implementations. We describe ORGODEX, a model and methodology for engineering scalable RBAC implementations where hundreds or thousands of employees require access to information on a need to know basis to meet the responsibilities of their role within the organization. The aim of this dissertation is met during three distinct phases:

- In chapter 4, we refine our contributions, introducing and motivating the requirement for new RBAC structural relationships that distinguish between roles and responsibilities, thereby satisfying our first objective.
- In this chapter, we introduce our new ORGODEX model and methodology for engineering scalable RBAC implementations, thereby satisfying our second and third objectives. We use our operational case study at ACME university to validate the utility of our new model and methodology.
- In chapter 6, we further validate the general applicability of ORGODEX, leading a one year project to extend features and functionalities of the ACME university information system to a partner institution.

ORGODEX is founded in RBAC, assimilating decades of research to realize a hybrid RBAC-ABAC role-centric model, addressing several open problems described for the core ABAC model [78][34][39]. This includes the lack of an agreed upon or foundational model and limited emulation for representing more mature models like RBAC. This is an important concern for organizations that have significant investments in large RBAC implementations where hundreds of roles have been engineered to realize security policy and safeguard information assets for several years if not decades.

ORGODEX offers a practical alternative for large organizations who do not see value in redesigning their information security architecture to use the ABAC model and its eXtensible Access Control Markup Language (XACML) policies. Unlike traditional access control technologies, such as RBAC, that have a proven track record in being adopted in large scale real-world systems, ABAC is still largely unproven in terms of practical scalability [78].

In the following sections, we produce a new granular, evolutionary model and methodology for describing and reasoning about RBAC implementations, using roles and responsibilities to directly inform the architecture. First, we introduce ORGODEX, our new model for describing and reasoning about RBAC implementations using the terms role and responsibility to facilitate communication. We explain why a mandatory separation between roles and responsibilities permits engineers to more easily communicate with one another and we stress the importance of communication with the business side of an organization.

Next, we deliver our new methodology for engineering scalable implementations, using our ORGODEX model to iteratively analyze, realize and publicize a RBAC implementation. We use the ISO/IEC 27001 standard for specifying information security management systems, adopting the Plan-Do-Check-Act (PDCA) cycle for continuous improvement, we describe how ORGODEX facilitates communication, delivering consistent access control solutions that directly reflect the business model. We validate the utility of our methodology by reverse engineering the ACME university example from the previous chapter.

Finally, we discuss how ORGODEX saves time for organizations. Unlike the classic triangular RBAC model where the savings are easily demonstrated, with ORGODEX we make the case for investing minutes to save hours, days and weeks when maintaining large RBAC implementations over a span of years and across decades.

At the conclusion of this chapter, the reader should better understand why the ORGODEX model and methodology saves time for organizations with large RBAC implementations, recognizing the added value of working away from the classic triangular model and towards a granular, evolutionary model where roles and responsibilities have semantically diverged to improve communication, thereby lowering the cost of maintenance.

5.2 Background

From the year 2000 onwards, we have analyzed, realized and publicized the RBAC implementation of ACME university, resulting in a deep-seated appreciation of the inherent challenges faced when administering RBAC across decades. We have closely observed and responded to the impact of employee turnover and security policy changes in a real-world setting, forming the motivation for this research and influencing our work.

The University Information System (UIS) is an Oracle^{TM}database containing hundreds of tables designed by a small team of IM specialists responsible for software development, enhancement and integration in support of both national and international clients. In the late 1990's this team consisted of fewer than five members. Today there are about a dozen employees responsible for supporting the information needs of multiple learning institutions, thousands of students, hundreds of staff members and an eclectic group of affiliates, including hundreds of contractors.

As RBAC practitioners, we have found it extremely challenging to maintain RBAC implementations that are tightly coupled with security policy. In our experience, the complexity of RBAC implementations is not well appreciated. As RBAC researchers, we are well aware of the popularity of the RBAC model despite its open, generic framework for engineering solutions that may be costly to maintain [12] [52] [58]. In this dissertation, we are motivated to convince the reader that the implicit subject-permission model hides too many important details to the detriment of RBAC practitioners, especially those accountable for administering hundreds or thousands of roles.

In the following section, we produce a new model for describing and reasoning about RBAC implementations using roles and responsibilities to directly inform the architecture, publishing explicit, comprehensible representations of RBAC implementations that are more easily understood and receptive to change. The reader should not confuse the ORGODEX model with the Open Authorization (OAuth) solution re-purposed for user authentication by major identity providers such as Facebook, Google and Microsoft [11]. Where OAuth 2.0 defines an open framework for access control delegation, typically permitting users to share information about their accounts with third party applications or websites, ORGODEX is a model and methodology for engineering scalable access control systems [80].

To aid comprehension, equating OAuth and ORGODEX is like comparing insurance companies and manufacturers in the automobile industry. OAuth is like an insurance company where the business problem requires individuals to share their personal information to establish trust relationships and accountability whereas ORGODEX is like an automobile manufacturer where the objective is determining how to make a better automobile given what we know today and what we believe will be true in the future.

5.3 Model

In business management a roles and responsibilities matrix may be used to clarify who is responsible for what [30]. A mandatory separation between roles and responsibilities is an evolutionary approach to implementing RBAC, permitting engineers to more easily communicate the important aspects of their design implementations to one another. In addition, describing and reasoning about RBAC implementations using the terms role and responsibility facilitates communication with the business side of an organization, allowing practitioners to publish comprehensible security information. In this section, we introduce ORGODEX, our new model (and methodology) for engineering scalable RBAC implementations in large organizations where employees require access to information on a need to know basis.

Figure 5.1a is an intentional repetition of Figure 2.6b meant to highlight the similarities and differences between RBÄC and ORGODEX as follows:

- ORGODEX discards the implicit SP relation
- S is renamed **Workforce**
- SR+ is renamed **WHO**
- R+(RH+) is renamed **Role**
- G is renamed **HAS ACCESS TO**
- D+(DH+) is renamed **Responsibility**
- PD+ is renamed **WHAT**
- P is renamed **Information**
- ORGODEX introduces **CONSTRAINT(S)**

ORGODEX workforce-role-responsibility-information relationships are defined and enforced. There is **always** *a role and responsibility* between the workforce and information. In Figure 5.1b, we see that the ORGODEX model builds upon RBÄC and the notion of role evolution, using business terminology and providing support for role-centric constraints.

Definition 20. The **Workforce** is enrolled into roles. Dr. George Scott is a member of the workforce fulfilling the DC #1 role.

Definition 21. The ability to review and maintain **Information** is assigned to a responsibility. A DC, responsible for approving grades, requires access to course information to meet this responsibility.

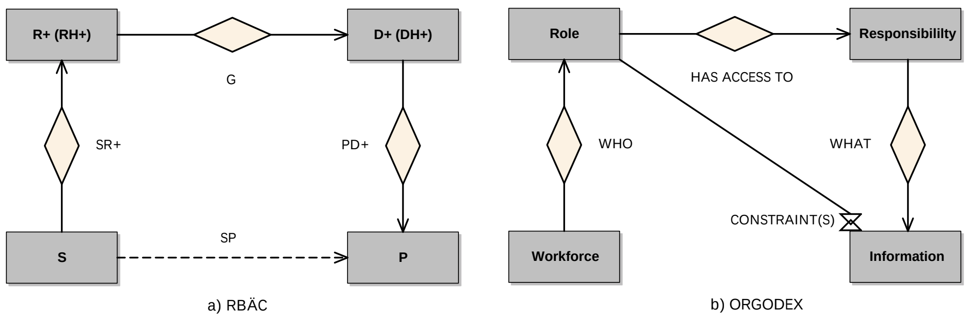

Figure 5.1: The ORGODEX model builds upon RBÄC and the notion of role evolution, using business terminology and providing support for role-centric constraints.

Definition 22. An **Attribute** has a name and holds a value. For example, the (name,value) tuple (Department,#1).

Definition 23. A **Constraint** limits information accessible for role-responsibility relationships. A DC, responsible for approving grades for (Department,#1), requires access to course information for Department #1 only when approving grades.

ORGODEX retains the principal semantic domains underlying RBAC and RBÄC, adding constraints and using business terminology:

- Let W be a Workforce, a set of subjects, s
- Let I be Information, accessible through a set of permissions, p
- Let R^o be a set of roles defined in an organization
- Let R^e be a set of responsibilities defined in an organization[1]
- Let A be a set of attributes defined as (name, value) tuples in an organization
- Let $C \subseteq R^o \times R^e \times A$ be a role-responsibility-information constraint relation
- Let $WR^o \subseteq W \times R^o$ be a workforce-role enrollment relation
- Let $IR^e \subseteq I \times R^e$ be an information-responsibility assignment relation
- Let $R^oH \subseteq R^o \times R^o$ be a role-role-hierarchy relation, R^oH is required to be acyclic
- Let $R^eH \subseteq R^e \times R^e$ be a responsibility-hierarchy relation, R^eH is required to be acyclic

[1] R^o and R^e are disjoint sets

- Let $\mathrm{R^oR^e} \subseteq \mathrm{R^o} \times \mathrm{R^e}$ be a grant relation

In terms of complexity, Kuijper and Ermolaev provide a proof that RBAC is equivalent to RBÄC.

Theorem 2. There exists a linear translation from RBAC to RBÄC and, vice versa, there exists a linear translation from RBÄC back to RBAC [46].

RBÄC defines the following syntax:

- Let R+ be a set of proper roles
- Let D+ be a set of demarcations[2]

Considering the trivial case of translating ORGODEX to RBÄC, we arrive at the following theorem.

Theorem 3. There exists a linear translation from RBÄC to ORGODEX and, vice versa, there exists a linear translation from ORGODEX back to RBÄC.

Proof. We provide a constructive proof by mapping each role ($\mathrm{R^o}$) to a proper role (R+) and each responsibility ($\mathrm{R^e}$) to a demarcation (D+).

- $\mathrm{R^o} = \{role[r] \mid r \in R{+}\}$
- $\mathrm{R^e} = \{responsibility[d] \mid d \in D{+}\}$
- $\mathrm{WR^o} = \{(s, role[r]) \mid (s, r) \in SR{+}\}$
- $\mathrm{IR^e} = \{(p, responsibility[d]) \mid (p, d) \in PD{+}\}$
- $\mathrm{R^oH} = \{(role[r], role[r\prime]) \mid (r, r\prime) \in RH{+}\}$
- $\mathrm{R^eH} = \{(responsibility[d], responsibility[d\prime]) \mid (d, d\prime) \in RD{+}\}$
- $\mathrm{R^oR^e} = \{(role[r], responsibility[d]) \mid r \in R{+},\ d \in D{+}\}$

Next we consider the case of translating ORGODEX to RBÄC, highlighting the semantic equivalence of these two models when not considering the extension for constraints defined by ORGODEX:

- $\mathrm{R+} = \mathrm{R^o}$
- $\mathrm{D+} = \mathrm{R^e}$
- $\mathrm{SR+} = \mathrm{WR^o}$
- $\mathrm{PD+} = \mathrm{IR^e}$
- $\mathrm{RH+} = \mathrm{R^oH}$
- $\mathrm{RD+} = \mathrm{R^eH}$

[2]R+ and D+ are disjoint sets

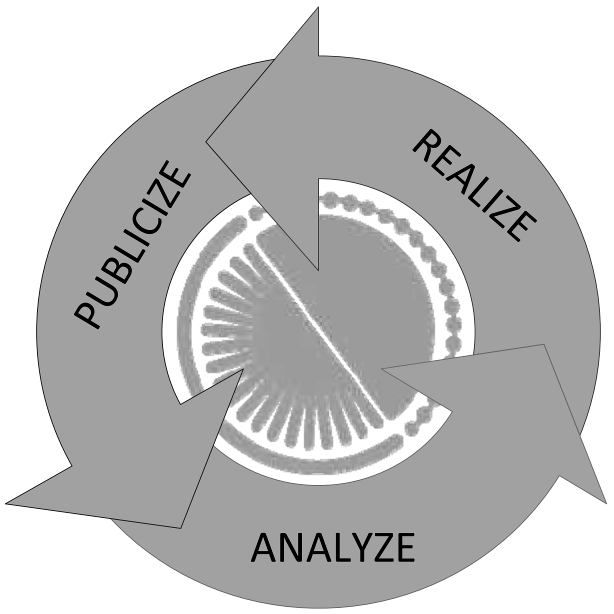

Figure 5.2: The ORGODEX methodology facilitates communication, providing a practical framework for consistent access control solutions through iterative analyze, realize and publicize cycles.

The requirements for scalable access control and management architectures presents a unique set of challenges. In terms of scalability, ORGODEX provides support for several languages and applications, permitting the use of multiple interfaces for configuring security policies described by roles, responsibilities and constraints. Moreover, ORGODEX facilitates decentralized management in distributed architectures where the administration of roles and responsibilities is just another responsibility, thereby providing cost effective maintenance without sacrificing efficiency [43].

In the following subsections, we describe the ORGODEX methodology for analyzing the business, realizing a RBAC implementation and publicizing the implemented security architecture.

5.4 Methodology

The ORGODEX methodology is based upon the ISO/IEC 27001 standard for specifying information security management systems, adopting the PDCA cycle for continuous improvement [35][21]. It facilitates communication by

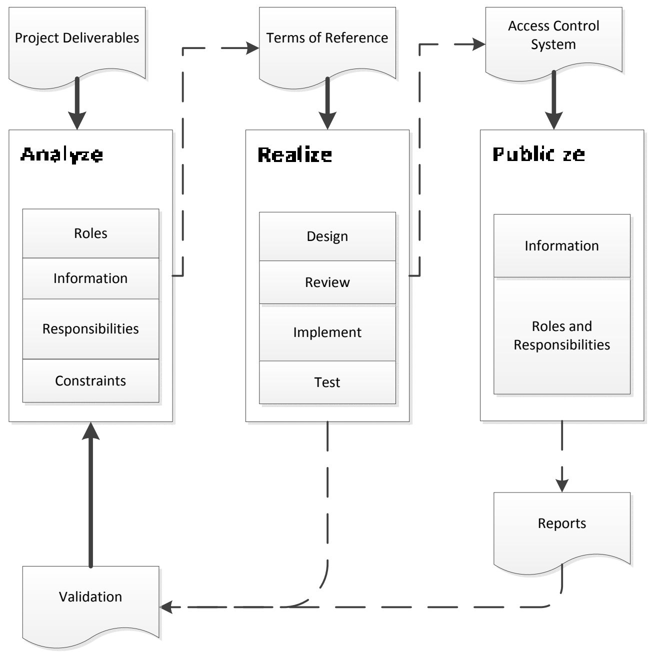

Figure 5.3: The ORGODEX model facilitates communication, providing a practical framework for complex, scalable RBAC implementations where the outputs of one phase supply the inputs to the next phase in continuous feedback loops.

providing a practical framework for consistent access control solutions through iterative analyze, realize and publicize cycles (Figure 5.2).

In Figure 5.3, the methodology begins with the Analyze phase where Project Deliverables serve as the input in this example. The initial input, or starting point, could also be a Statement of Requirements (SOR) or milestone or any work objective defined to meet a business outcome.

5.4.1 Analyze

The PDCA cycle begins with the Plan phase. With ORGODEX, the analyze phase may be completed by a Business Analyst (BA) with project deliverables serving as inputs and the following terms of reference produced as outputs:

1. Roles
2. Information
3. Responsibilities
4. Constraints

BAs are expected to elicit outputs by asking questions related to the project deliverables and using the five W's; who, what, why, when and where to ultimately produce a roles and responsibilities matrix.

According to Ziv's Law software development is unpredictable and documented artifacts such as specifications and requirements will never be fully understood [87]. For this reason and others, communication is often considered the most important aspect of Project Management, justifying the need to work towards shared goals, objectives and outcomes with the following assumptions:

- Limited information is available when planning and initiating a project
- No two stakeholders share the exact same imagined outcome

In this dissertation we focus on the added benefit of integrating role and requirements engineering efforts when designing system functions and user interfaces. We also identify candidate roles as nouns and similar to the methodology described by Coyne [13], we identify responsibilities by determining activities that may be stated as a verb-information pair, for example *Approve Final Grades.* Finally, we link up roles and responsibilities, identifying role-centric constraints when applicable.

Roles ($\mathbf{R^o}$)

Requirements Engineering (RE) is not only a process of discovering and specifying requirements, it is also a process for facilitating effective communication among different stakeholders. RE papers generally note that access control is often considered late in the software development cycle and consequently a thorough validation of the security requirements is not performed. Many authors suggest that access control policies must be elicited early in the development cycle, the RE phase [17][32][33]. In their field study, Wilander and Gustavsson discover that 75% of security requirements are functional, leading them to conclude that following *well-known and rigorously reviewed standards* would make the management of security features equivalent to other functional requirements.

Haley, Moffett, Laney and Nuseibeh describe a framework for security requirements engineering [32]. From RE they utilize the concept of operationalized functional requirements with appropriate constraints. From security engineering they use the concept of assets and deal with threats to those

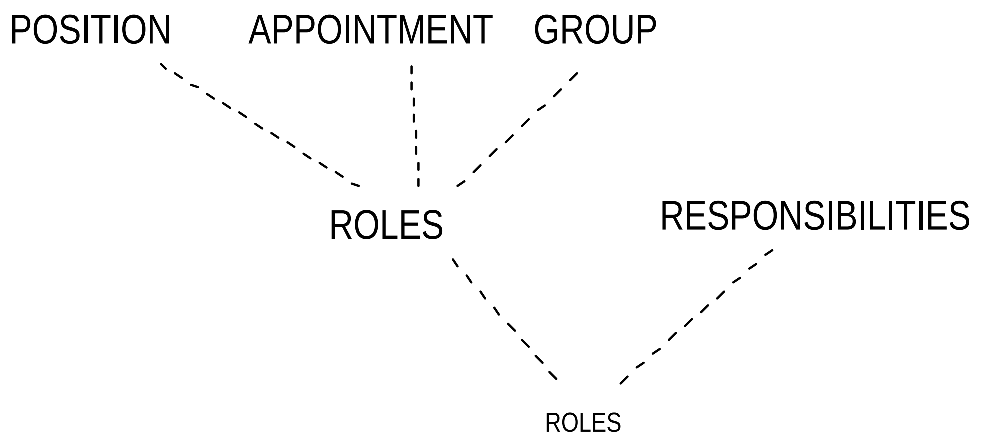

Figure 5.4: At ACME university, Role Evolution diverges beyond Roles and Responsibilities.

assets. He and Antón prescribe a method for deriving access control policies from Software Requirements Specifications (SRS) and database designs. In addition, He and Antón have published a goal-driven framework for bridging the gap between high-level privacy requirements and low-level access control policies [33]. Crook, Ince and Nuseibeh relate roles to organizational theory and describe how these roles can be employed to define access policies [17].

Like Crook et al. [17], we believe that roles must be typed. Role typing is an important aspect of the role analysis process. In Figure 5.4, we illustrate how roles have diverged beyond roles and responsibilities at ACME university where appointment, position and group roles are identified, providing further evidence of an organic role phylogeny that is occurring in real-world RBAC systems. In the context of ACME university, we note the following:

- A position is a role typically held by an employee on an indeterminate, term or casual basis. A position is associated with a number and title in the Human Resources Management System (HRMS), for example, position number #123456, Project Manager.
- An appointment is a role that one fulfills for a specified (or unspecified) period of time above and beyond the responsibilities of a staffing position, for example, a Departmental Chair.
- Logically, there are sets of roles that are grouped because they share similar responsibilities. In many cases this is done to distribute the workload. For example, DC#1 and DC#2 require access to different sets of information.

Table 5.1: A Role Analysis Document identifies WHO, establishing a relationship between the Workforce and a defined Role.

R^o	Type	Description
DC#1	Appointment	Departmental Chair #1
DC#2	Appointment	Departmental Chair #2
DC	Group	Departmental Chair

To build an instance of ORGODEX, an analysis begins by identifying WHO the information management system and its security will support, analyzing the roles to better understand the workforce. At ACME university, fourteen Departmental Chairs report to one of three Deans. In Table 5.1 we produce a small sampling of a role analysis document. We do not identify members of our workforce (e.g. Dr. George Scott), instead we focus on the roles they fulfill at ACME university.

It is important for the reader to consider the impact of this trivial exercise and the rationale for its placement as the lead output of the ORGODEX analyze phase. From the onset, prior to any development activities the project team collaboratively produces a simple document (Table 5.1) identifying the intended workforce (W) and their respective roles (R^o), potentially redefining the scope of the project with the shared realization that the workforce is larger or smaller than anticipated. Furthermore, a simple tool for better understanding roles and their maintenance, including addition and subtraction is quickly established.

Information (I)

ORGODEX separates WHO (R^o) and WHAT (I) from the onset, mandating that BAs identify terms of reference and associate protection levels as follows:

- The flow of information is primarily orchestrated around notional concepts such as students and courses
- Protected information may only be viewed by those who have been authorized
- Public information is not protected and may be published for all to see

In Table 5.2 we produce a small sampling of an information analysis document, highlighting concepts of interest such as Course and Student. The rationale for determining the protection levels early in the analysis process is twofold. First, we must establish a balance between security and practicality from the onset. Information deemed to require additional protection levels

Table 5.2: An Information Analysis Document defines Terms of Reference including whether or not the information is deemed Protected or Public (Yes or No)

Information (I)	**Description**	**Protected**
Student	Identifies individuals engaged in the pursuit of higher education	Yes
Course	Identifies the subject and learning objectives to be delivered in a defined period of time (e.g. English 101)	No

must be further considered upstream (i.e. Responsibility Analysis). Second, we observe that personnel at ACME university are often unsure of the protection level for the information they are responsible for, often unknowingly compromising security protocols. To this end, we do not suggest our methodology will fix this problem, however, we do believe that the last phase of our methodology, publicize, should link back to these protection levels, advertising information security on a regular basis, ideally strengthening the posture of an organization.

Responsibilities (R^e)

ORGODEX next prescribes the identification of responsibilities, intentionally separating WHO (R^o) and WHY (R^e) from the onset. The objective of a responsibility analysis document is to promote communication amongst project team members early and often, working towards a shared understanding of the system-to-be and prioritizing requirements, ensuring that they can be read, analyzed, (re-)written, and validated [56].

BAs use requirements documentation to produce a simple three column table including the following fields:

- Information: A relationship to our definition of WHAT (I) .
- Responsibility: The focus of this document, identifying responsibilities (R^e).
- Requirement: A relationship to the features and functionalities identified in requirements documentation.

BAs must analyze responsibilities independently from those charged with their completion. We contend that the responsibility analysis document may

Table 5.3: A Responsibility Analysis Document further identifies WHAT, including the related information concept, a novel high-level responsibility and the initiating requirement.

Information (I)	**Responsibility (R^e)**	**Requirement**
Course	Review Course Information	View list of students associated with course offering
Student	View Final Grades	Review results achieved by students in a course offering
Student	Approve Final Grades	Approve results achieved by students in a course offering

be one of the most important work products established within our methodology. We suggest this based on years observing enterprise organizations where responsibilities dynamically shift on a daily, weekly, monthly and yearly basis. From an information management security viewpoint, we contend that this is the fundamental difference with our model. In Table 5.3 we list records of interest from a responsibility analysis document, describing the information-responsibility relationship.

Consider the impact of this ORGODEX work product and its timing. From the onset, prior to any development activities the project team collaboratively produces a simple document (Table 5.3), identifying the Information (I) assets and the related responsibilities (R^e), collaboratively working towards a shared understanding of the system-to-be and its security architecture. Furthermore, the maintenance, including addition, modification and subtraction of responsibilities is established in a very short period of time by determining WHY (R^e) access to information is required and with the understanding that WHO (R^o) requires access is subject to change in both the short and long term.

Constraints (C)

Finally, ORGODEX mandates a constraints analysis before linking roles and responsibilities, formalizing WHEN (C) and WHERE (C) access to information is required. The result of a constraints analysis exercise is a roles and responsibilities matrix, identifying WHO - HAS ACCESS TO - WHAT, indicating WHY, WHEN and WHERE as applicable. During this activity, the BA must analyze the constraints or scope for each responsibility. Although this is all very abstract and listed sequentially, it is assumed these activities and their related work products will be carried out simultaneously using var-

Table 5.4: A constraints analysis document identifies WHO - HAS ACCESS TO - WHAT, indicating WHY, WHEN and WHERE as applicable.

WHO (R^o)	**WHAT (I)**	**WHY (R^e)**	**WHEN (C)**	**WHERE (C)**
DC#1	Course	Review Course Information	Course is administered by DC#1	UIS
DC#2	Course	Review Course Information	Course is administered by DC#2	UIS

ious requirements elicitation techniques, perhaps interview questions similar to those described by Jaferian [38].

The roles and responsibilities for a Departmental Chair has been a subject of interest within the research community [8]. At ACME university, each and every DC shares similar if not identical permissions but they are responsible for different information sets. During our constraints analysis, we observe DCs are responsible for approving final grades for students taking a course delivered within their respective department. Grades are entered by instructors who teach an academic course to students, set up an evaluation scheme and grade the student. In Table 5.4 we highlight the roles and responsibilities of interest for this chapter, indicating that DC#1 is only responsible for reviewing department #1 course information. Similarly, DC#2 is only responsible for approving final grades for courses offered by department #2.

In the following subsection, we describe how the work products of the BA, listed in this section, are directly leveraged during the realize phase.

5.4.2 Realize

The PDCA cycle continues with the Do and Check phases. With ORGODEX, the realize phase may be completed by a Security Architect (SA) using the terms of reference defined by BAs during the analyze phase as inputs to engineer an access control system using the following highly iterative process:

1. Design
2. Review
3. Implement
4. Test

A version of Humphrey's Law states that *users don't know what they want until they see working software* while another version suggests *people don't know what they want until they see what they don't want.* Agile methodologies attempt to put the software being developed first, acknowledging that user requirements will change both during and after projects [36]. ORGODEX embraces the following agile principles:

- Deliver working software frequently
- Business people and developers must work together daily
- Working software is the primary measure of progress
- Promote sustainable development

In the following subsections we present our methodology for realizing information security. We elaborate upon our scenario of interest from the previous chapter where DC#1 and DC#2 are responsible for approving final grades at the end of each academic term.

Design

During the Design phase, the SA uses terms of reference from the analyze phase as inputs to create independent roles and responsibilities, proposing hierarchies where applicable. In Figure 5.5, the appointment roles DC#1 and DC#2 are enrolled in the group role DC forming a role-role hierarchy relation, R^oH. Similarly, the responsibilities *Approve Final Grades* and *View Final Grades* form a responsibility-responsibility hierarchy, R^eH.

Review

During the Review phase, the SA utilizes the ORGODEX hierarchical diagramming technique to validate their comprehension of the business model, using a visual representation of the directional hierarchy of workforce-role, role-role, role-responsibility, responsibility-responsibility and responsibility-information relationships [20]. Once validated, the design is used by the SA to implement the access control system.

Implement

First, with a reviewed list of roles (Table 5.1), the SA uses the CREATE ROLE command three times as follows:

- CREATE R^o DC#1;
- CREATE R^o DC#2;
- CREATE R^o DC;

Then based upon design reviews (Figure 5.5), the SA implements role hierarchies (R^oH), issuing the following commands:

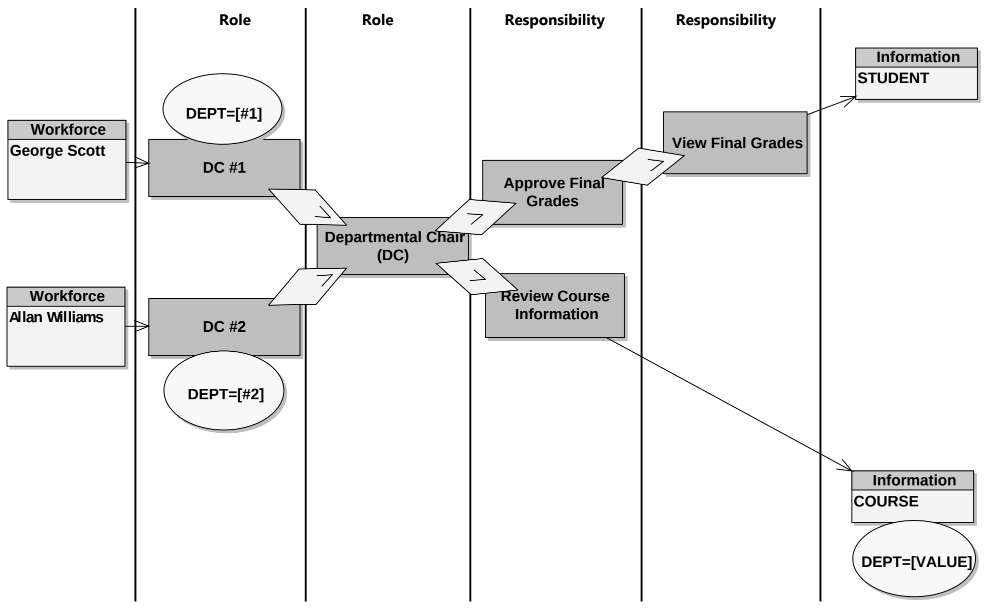

Figure 5.5: The hierarchical diagramming technique employed by ORGODEX may be used by a Security Architect to validate their comprehension of the business model.

- CREATE R^{o}H between DC#1 and DC;
- CREATE R^{o}H between DC#2 and DC;

Similarly, with a defined set of responsibilities (Table 5.3), the SA would use the CREATE RESPONSIBILITY command three times as follows:

- CREATE R^e *REVIEW COURSE INFORMATION*;
- CREATE R^e *VIEW FINAL GRADES*;
- CREATE R^e *APPROVE FINAL GRADES*;

Then based upon design reviews (Figure 5.5), the SA would implement a responsibility hierarchy (R^{e}H), issuing the following command:

- CREATE R^{e}H between VIEW FINAL GRADES and APPROVE FINAL GRADES;

Creating independent roles and responsibilities allows the SA to initiate a simple framework for application developers to assign permissions to independent responsibilities based on the functional requirements without being overly concerned about who will ultimately hold the responsibility.

Next, we must add constraints where A is a set of (name,value) pairs and C is a set of role-responsibility-information constraint relationships. In the following example, we add ORGODEX role-centric constraints, permitting DC#1 and DC#2 to review course information for their respective departments:

- ADD A_1 = (DEPT, #1)
- ADD A_2 = (DEPT, #2)
- ADD C (DC#1, REVIEW COURSE INFORMATION, A_1)
- ADD C (DC#2, REVIEW COURSE INFORMATION, A_2)

Constraints are applied by creating database views with support for ORGODEX role-centric constraints. For instance, #1 is substituted for [value] in the following database view when DC#1 is reviewing course information:

- SELECT information FROM course WHERE dept in [value])

It is important for the reader to note that unless constraints are defined in this example, no records are returned for members of the workforce accessing this database view, imposing a strong default security posture.

Finally, the SA links up roles and responsibilities using the following GRANT commands:

- GRANT *APPROVE FINAL GRADES* to DC;
- GRANT *REVIEW COURSE INFORMATION* to DC;

Test

During the Test phase, the SA grants the appointment roles to the applicable workforce members in the development system, ensuring that features and functionalities are working to satisfaction before coordinating a review and approval demonstration session for concerned stakeholders. For instance, the SA expects DC#1 to only have access to their respective courses (Figure 5.5).

- GRANT DC#1 to George Scott;
- GRANT DC#2 to Allan Williams;

Following a review and approval demonstration, the SA would use the feedback obtained to complete the remaining work before granting roles to workforce members in the production RBAC implementation.

5.4.3 Publicize

One revolution of the PDCA cycle completes with the Check and Act phases. In the ORGODEX publicize phase, the realized RBAC implementation is used to produce the following reports at a minimum:

1. Information
2. Roles and Responsibilities

According to Conway's Law, software architecture tends to mirror the designing organization. Kwan suggests this law is important when developing software from a responsibility or task-level perspective [47]. Using roles and responsibilities, ORGODEX mirrors the communication structures of an organization, directly reflecting its business model and implicitly coordinating the distributed development of secure information management systems [65].

During the publicize phase, we report the realized roles and responsibilities, facilitating their ongoing validation with terms of reference that directly reflect the business model. The intent is to display real security architecture information on demand for authorized members of the workforce using a comprehensible roles and responsibilities matrix to regularly validate profiles [6]. With ORGODEX this is just another responsibility (R^e) that may be constrained (C).

Information

Over time a business develops and operates using a unique glossary or terms of reference. Some terms might be well understood concepts like student or course while other terms may be very context specific. For this reason, it is important for large organizations to develop and maintain searchable terms of reference that are regularly validated as part of the ORGODEX publicize cycle.

The Information Asset Report (Table 5.5) serves as an operational glossary of terms that are to be used by BAs, SAs and Developers to elicit and confirm requirements both within and between distributed organizations [65]. These terms are communicated out to the organization.

Roles and Responsibilities

The Roles and Responsibilities Report (Table 5.6) identifies WHO - HAS ACCESS TO - WHAT, indicating WHY, WHEN and WHERE as applicable:

- The report is an inclusive matrix listing roles and their assigned responsibilities.

Table 5.5: The Information Asset Report is subjected to ongoing validations, publishing the Terms of Reference including whether or not the information is deemed Protected or Public (Yes or No)

Information	**Description**	**Protected**
Student	Identifies individuals engaged in the pursuit of higher education	Yes
Course	Identifies the subject and learning objectives to be delivered in a defined period of time (e.g. English 101)	No

Table 5.6: The Roles and Responsibilities Report is subjected to ongoing validations, publishing WHO - HAS ACCESS TO - WHAT and dicating WHY, WHEN and WHERE as Applicable.

WHO R^o	**WHAT (I)**	**WHY (R^e)**	**WHEN (C)**	**WHERE (C)**
DC#1	Course	Review Course Information	Course is administered by DC#1	UIS
DC#2	Course	Review Course Information	Course is administered by DC#2	UIS

- The report resembles a job description for individual roles

Communication with the business side of an organization is a primary motivation for ORGODEX. We believe that the Roles and Responsibilities Report in particular is an interesting solution to the challenging problem of entitlements review where managers are expected to validate the permissions held by their subordinates [38]. If this report directly reflects the work produced by the BA during the Analyze phase, we believe a communication language that crosses from the business world to the information security world has been achieved. One can envision a software application whereby managers regularly review an interactive Roles and Responsibilities report for their employees, determining whether their access is too restrictive or overly permissive [5].

With ORGODEX the intent is to report security architecture information both on-demand and periodically, utilizing a comprehensible roles and responsibilities matrix to regularly validate profiles. Although we do not focus upon the notion of *Insider Threat* in this work [10][35], we believe ORGODEX inherently addresses this concern, delivering a model and methodology based upon the ISO/IEC 27001 standard, where provisioning information on a *need-to-know* basis is a primary objective.

In the following section, we discuss how the ORGODEX model and methodology controls growth, using role evolution to *insulate* RBAC systems against ongoing employee turnover, policy changes and reorganization.

5.5 Discussion

It is surprising that the business model of large organizations is often loosely coupled with its RBAC implementations, merging the notions of role and responsibility into simplistic, inflexible subject-role-permission relationships. Perhaps this is directly linked to the classic RBAC example where all bank tellers share the same role and have the exact same responsibilities. Unfortunately, this example oversimplifies RBAC, making it difficult to understand the challenges inherent when implementing RBAC in large dynamic organizations with highly diversified workforces.

ORGODEX discards the implicit subject-permission relation. This may seem trivial but the intent is quite deliberate. Implicit SP relationships hide too many details, trivialize RBAC implementations and impede communication. Managing roles, subjects and their interrelationships is a formidable task that is often highly centralized in small teams of administrators [74]. A daunting task in large organizations where access control is a secondary duty resulting in considerable misunderstanding amongst practitioners [9]. Instead with ORGODEX, we have an explicit roles and responsibilities matrix produced by BAs, implemented by SAs and regularly reviewed by those responsible for validating their employees access to information.

Recall from chapter 1 that with RBAC the administrative savings are easily demonstrable (Figure 5.6). Using our scenario of interest from ACME university, the SA creates the role DC, assigns the permissions to access five database views to DC and then enrolls each subject into the role DC. By creating the role DC the SA invests two extra administrative actions, however, the return on investment is four fewer administrative actions for each additional subject enrolled into the *DC* role, a savings of fifteen administrative actions (less two) for five DCs.

Unfortunately, this example oversimplifies RBAC, making it difficult to understand the challenges inherent when implementing RBAC in large dynamic organizations like ACME university where thousands of access control relationships must be maintained by a small team of SAs. At ACME university, there are a total of eleven DCs, conservatively each has five responsibilities

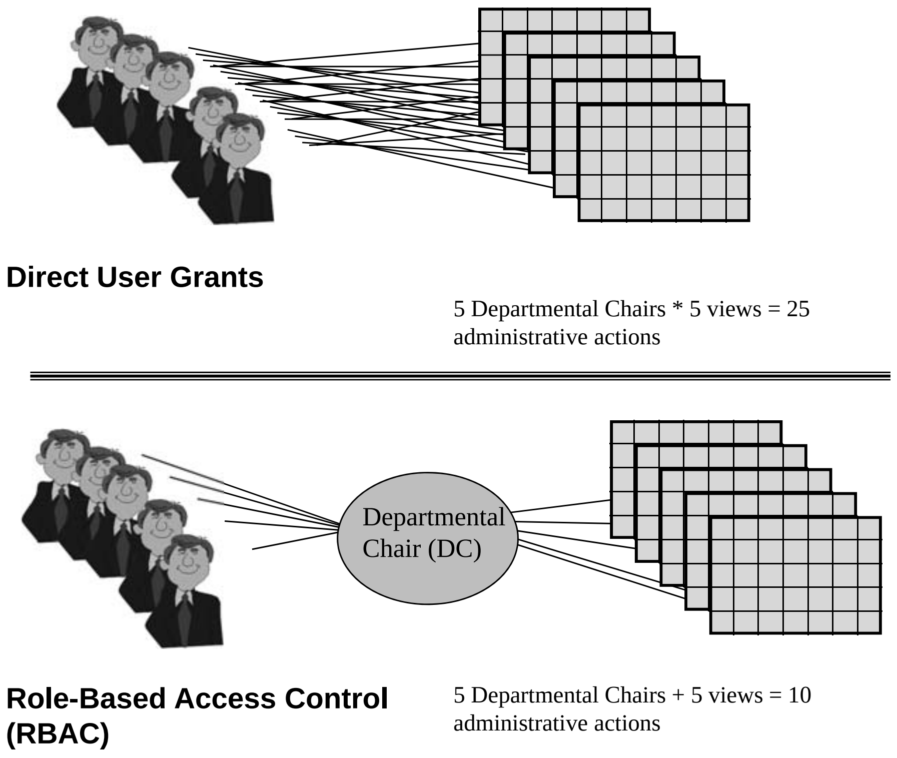
Direct User Grants
5 Departmental Chairs * 5 views = 25 administrative actions
Departmental Chair (DC)
Role-Based Access Control (RBAC)
5 Departmental Chairs + 5 views = 10 administrative actions

Figure 5.6: ACME Example of RBAC Administrative Savings

defined, each responsibility has the permission to access at least two database views (Figure 5.7) for a total of ten permissions. In this real-world scenario the savings is calculated by determining the Cartesian product of multiplying the set of eleven DCs by ten database views for a total of 110 direct user grant relationships.

Simply creating the role DC and assigning ten permissions affords the SA a net savings of 87 direct SP relationships:

- 11 DCs multiplied by 10 database views is 11 * 10 = 110 administrative actions when performing direct user grants
- 11 DCs plus 10 database views is 11 + 10 = 21 administrative actions for an RBAC implementation
- This implies a savings of 110 - 21 = 89 administrative actions

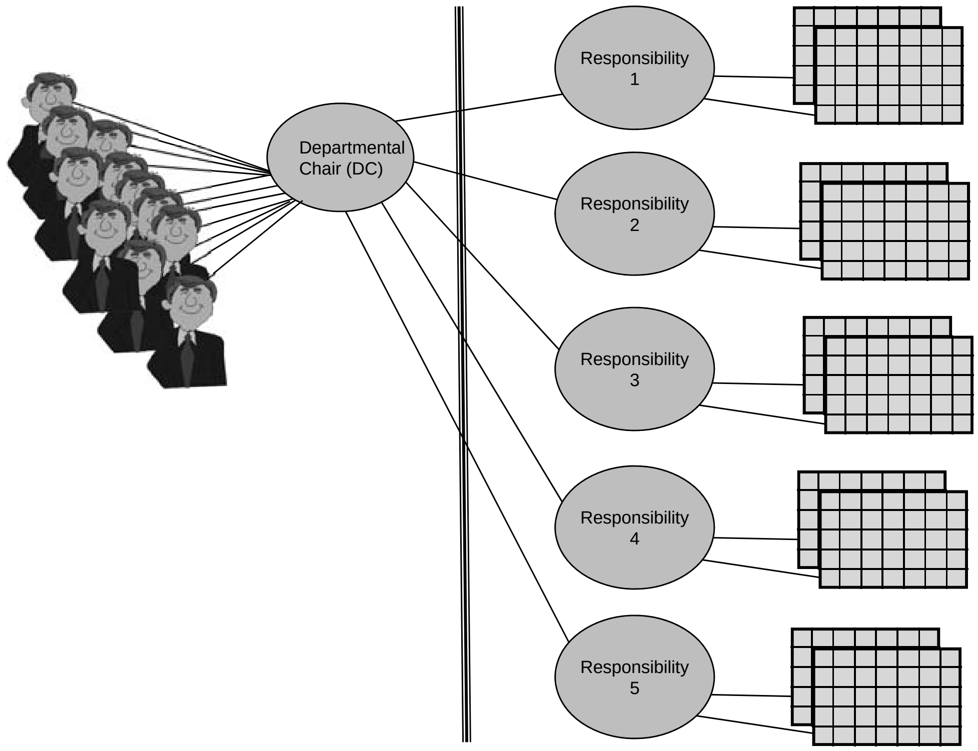

Figure 5.7: ACME Roles and Responsibilities Example

- Less the investment in one role creation and one role enrollment results in a savings of 89 - 2 = 87 administrative actions

ORGODEX and role evolution proposes further investment when introducing a responsibility. In Figure 5.7 the SA invests 10 additional administrative actions to create 5 responsibilities (R^e) and 5 role-responsibility grants (R^oR^e), for example (DC,*Responsibility #1*). The net savings is still 77 direct SP relationships.

The ORGODEX methodology invests extra administrative actions when defining responsibilities to describe and reason about RBAC implementations using business terminology. Unlike classic RBAC, the administrative savings are not easily demonstrable. The motivation for this design decision is best summarized as follows:

- access control may be delegated and managed in a roles and responsibilities matrix

- application architects can focus on mapping the permissions associated with functional requirements to responsibilities
- The degree of delegation for responsibilities is an interesting metric. The SA is not generally authorized to assign responsibilities absent the confirmation of an approving authority. Developing RBAC systems that facilitate entitlements review and permit managers to directly assign responsibilities to roles has the potential to impact the perception of access control.

Figure 5.7 depicts how each of these motivations are supported in a granular role-responsibility design as follows:

- If DC#1 chooses to delegate *Responsibility #1* in its entirety to their Administrative Officer, (AO#1), the result is one additional role-responsibility grant, ideally performed by DC#1 directly. In building a granular RBAC implementation where responsibilities are clearly defined we facilitate delegation. Furthermore, the result of this action is immediately apparent in the next publication of the Roles and Responsibilities Report for AO#1.
- If ACME university would like to develop functionality for a new responsibility (e.g. #6), application architects simply map permissions to the newly created *Responsibility #6*. For instance, in the development environment, *Responsibility #6* might be granted to DC during test only to later discover that this new responsibility will also be required by Deans and their Administrative Officers. With ORGODEX, responsibilities facilitate ongoing change.

Figure 5.7 provides a clear demarcation between the group role DC, and five responsibilities. As per the advantages listed above, the administrative savings are measured downstream in terms of adding new functionality (i.e. responsibilities), delegating and/or sharing a responsibility and finally when reporting the entitlements of an employee in a readable roles and responsibilities matrix.

Consider the alternative classic triangular model pictured in Figure 5.8. In this example we show how new functionality and its related permissions are often added for RBAC implementations. In contrast to the previous figure, permissions are added in sets of two, absent the definition of five responsibilities. The short term gain is minimal, likely measured in minutes. For each responsibility, the SA saves two administrative actions by not creating the responsibility and associating it with the role DC. In all likelihood the traceability for each additional set of permissions is not well-documented. This

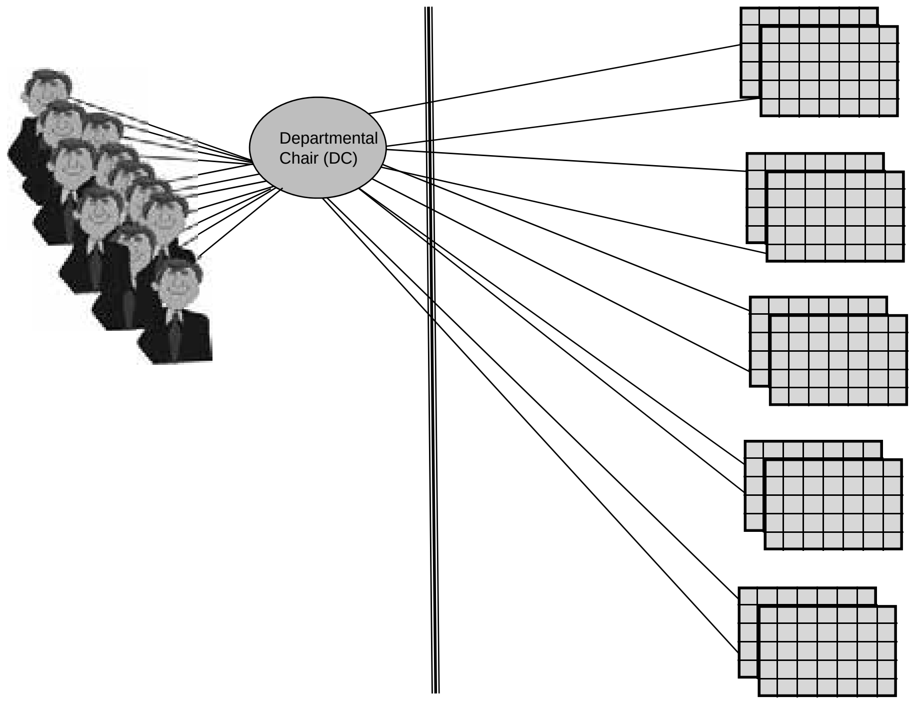

Figure 5.8: ACME Roles without Responsibilities Example

trivial time savings results in the loss of hours downstream when six months (or three years) later a responsibility is to be delegated or shared as follows:

- If DC#1 chooses to delegate *Responsibility #1* in its entirety to their Administrative Officer, (AO#1), the result is likely a support request that eventually finds its way to the SA who is now responsible for interpreting the requirement. After analyzing the permission set of DC to determine the required two permissions to perform *Responsibility #1*, the SA might choose to grant both permissions to AO#1 directly, further propagating the problem. By not building a granular RBAC implementation where responsibilities are clearly defined and documented, delegation becomes challenging, often measured in hours if not days. Furthermore, the individual with the authority to delegate a responsibility may be waiting days or weeks for the SA to complete their request. Clearly, investing the

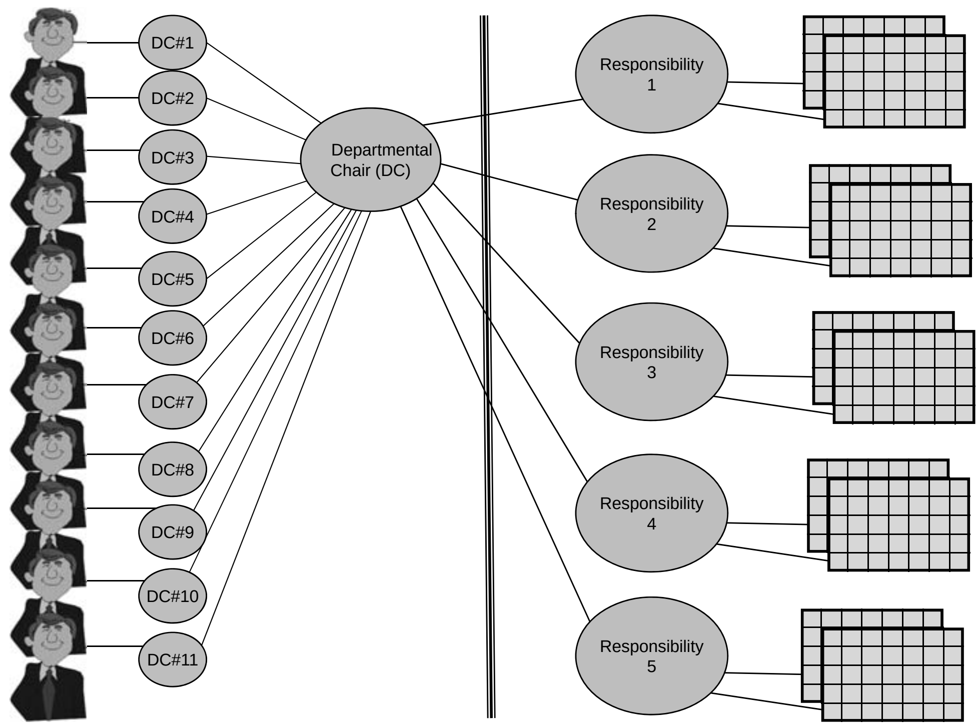

Figure 5.9: ACME Roles and Responsibilities Plus One Example

minutes required to introduce and document an ORGODEX responsibility produces administrative savings measured in hours and days based on this example. For businesses, the miscommunication and frustration are often palpable.

- If ACME university would like to develop functionality for a new responsibility (e.g. #6), application architects are unable to focus on mapping permissions directly to a newly created *Responsibility #6*. Instead, in the development environment, the permissions required to perform *Responsibility #6* are granted directly to DC. At deployment, the traceability for the two permissions required to perform this new responsibility are immediately lost, blending in to the accumulated permission set of DC.

ORGODEX and role evolution proposes further investment in roles and responsibilities. In Figure 5.9 the SA invests 22 additional administrative actions to create 11 constrained roles and 11 role hierarchies (R°H), for example (DC#1, DC). The net savings is still 55 direct SP relationships. The

ORGODEX methodology invests extra administrative actions when defining constrained roles to describe and reason about RBAC implementations using business terminology. This may seem counterintuitive to those whose beliefs are firmly entrenched in the classic triangular RBAC model. The motivation for this design decision is best summarized as follows:

- Workforce-role management may be automated as appropriate in organizations, reducing administrative overhead.
- Rather than producing metrics aimed at optimal role sets, we believe that the degree of automation for roles is of interest. At ACME university, the enrollment of employees into appointment roles is automated as part of existing business processes.
- Creating singular, multipurpose database views for various roles and responsibilities is desirable but expensive.

Figure 5.9 depicts how each of these motivations are supported in a granular role-role hierarchy design as follows:

- When a member of the workforce, for example Dr. George Scott, is appointed to DC#1 for a three year term, integrations with the HRMS may be leveraged to automate his enrollment into the role DC#1. If Dr. Scott takes an extended leave, for example a sabbatical, an acting appointment may be automated for another member of the workforce. While we acknowledge that the HRMS could likewise be used to automate the enrollment of DCs into the DC group role, this would not facilitate a fundamental aspect of the ORGODEX model (i.e. role-centric constraints).
- Automation aside, the administrative savings of the ORGODEX model and methodology is best exemplified when considering the addition of role-centric constraints and their application to database views. The creation of 11 constrained DC roles can be measured in minutes. In contrast, the creation and maintenance of singular, multipurpose database views may be measured in hours and days.

Figure 5.9 provides a clear demarcation between roles and responsibilities. As per the two advantages listed above the administrative savings are measured downstream when measuring the cost of maintenance. In opposition to the classic triangular model pictured in Figure 5.8, the short term gain is minimal, likely measured in minutes. For each of the 11 DC roles, the SA saves two administrative actions by not creating the role-role hierarchy relationships found Figure 5.9. The deficiency with this approach is the hours and days lost when asked to constrain a responsibility as follows:

- If a responsibility such as *Review Course Information* is to be constrained, application architects have several options. Creating 11 database views, one for each DC with a different constraining *where* clause is costly and infeasible because one would still need 11 DC roles to assign each of these new views to each DC
- Another popular option is ABAC. However, the shortcomings of this approach are described in detail in chapter 2. ORGODEX introduces role-centric constraints as a hybrid RBAC-ABAC model that does not rely on implicit relationships with business data, maintaining a security architecture that is abstracted and flexible

The challenge for ORGODEX is to demonstrate the administrative savings achieved by adding 11 role-role hierarchies for DCs. First, let's consider the net savings when comparing the classic triangular RBAC model (Figure 5.8) to the ORGODEX model and methodology (Figure 5.9):

- With the classic triangular model we save 87 administrative actions, both the investment and savings are easily demonstrated and measured in minutes. However, the long term cost of maintaining inflexible subject-role-permission triangles may be measured in hours, days and weeks.
- With ORGODEX, the net savings is 55 administrative actions. We invest 32 (i.e. 87-55=32) additional relationships at a cost of minutes to standardize the development of an architecture permitting large enterprise organizations to save hours, days and weeks.

The SAs at ACME university strongly contend their practical experience, explicitly separating roles and responsibilities, has proven to be an invaluable design decision when engineering a scalable RBAC implementation over the past two decades. The investment in *light-weight* roles and *explicitly defined* responsibilities results in a scalable, granular architecture where the payoff is much greater than the investment.

In our scenario of interest, the RBAC implementation includes two roles and one responsibility, permitting Dr. George Scott, DC#1, to *Review Course Information*. Although it may be difficult to see the immediate advantages of the ACME implementation, questioning the excessive number of roles and responsibilities, one begins to appreciate the simplicity of the subject-role relations found *on the surface* as they tunnel deeper into the design.

ACME university employs an evolutionary, fine-grained RBAC implementation where role-centric constraints are defined to directly influence the records returned in shared database views, for example:

- SELECT information FROM course WHERE dept = [value]

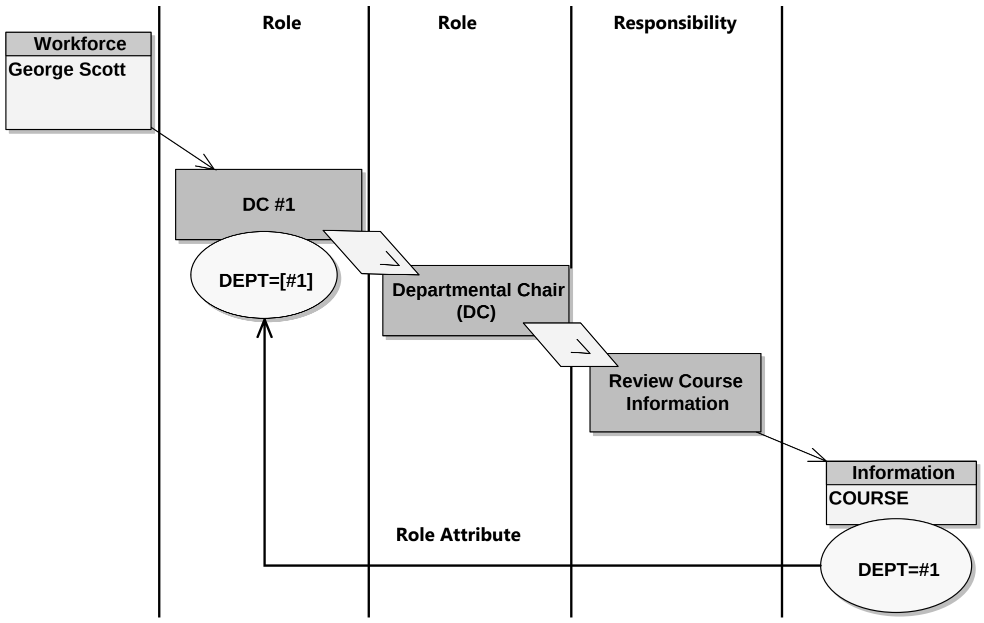

Figure 5.10: The List of Courses that may be selected by George Scott is constrained by the role-centric attribute DEPT with value #1.

- ADD A_1 = (DEPT, #1)
- ADD C (DC#1, REVIEW COURSE INFORMATION, A_1)

A database view is defined using Structured Query Language (SQL). In the example above all fields or columns are selected from the course table. When Dr. George Scott queries the database view, the role attribute (dept,#1) defined for DC#1 when reviewing course information is substituted for the parameter [value] at run time, thereby determining the set of courses that may be viewed by Dr. Scott.

Figure 5.10 depicts the *match* condition used to restrict the review of course information for Dr. Scott, fulfilling the role and responsibility requirement described in the first record of Table 5.7. For brevity, we do not include records for all eleven DCs. We do show how the database view is extended to support other groups of administrators in a scalable architecture that includes the AR-UG and AR-PG, briefly described in the previous chapter and further elaborated next.

In Figure 5.11 one can visualize how ACME university uses the flexibility and scalability of ORGODEX to create singular, multipurpose database views for tens if not hundreds of members of the workforce, providing both

Table 5.7: The Roles and Responsibilities Report may also be viewed for each Responsibility

WHO R^o	**WHAT (I)**	**WHY (R^e)**	**WHEN (C)**	**WHERE (C)**
DC#1	Course	Review Course Information	Course is administered by DC#1	UIS
DC#2	Course	Review Course Information	Course is administered by DC#2	UIS
AR-UG	Course	Review Course Information	Course is listed in Undergraduate Catalog	UIS
AR-PG	Course	Review Course Information	Course is listed in Postgraduate Catalog	UIS

constrained and consistent, context aware information to a variety of employees at the university. Combining the roles and responsibilities of these four members of the workforce in a hierarchical diagram results in a busy but explicit representation of the RBAC implementation. The intent of this example is not to confuse the reader but instead to help them better visualize the details of a real-world RBAC implementation found at ACME university.

It is important to realize that the Roles and Responsibilities Report may be viewed both by role and by responsibility as listed in Table 5.7. This facilitates entitlement review at ACME university. By decoupling roles and responsibilities and publicizing dynamic reports directly from the RBAC implementation, those charged with validations may be provided with simple reports indicating WHO, WHAT, WHY, WHEN and WHERE.

5.6 Summary

In this chapter we present our new granular, evolutionary approach for engineering RBAC implementations, making the following contributions:

- We introduce ORGODEX, our new model for describing and reasoning about RBAC using the terms role and responsibility to facilitate communication
- We deliver our new ORGODEX methodology for iteratively analyzing, realizing and publicizing scalable RBAC implementations

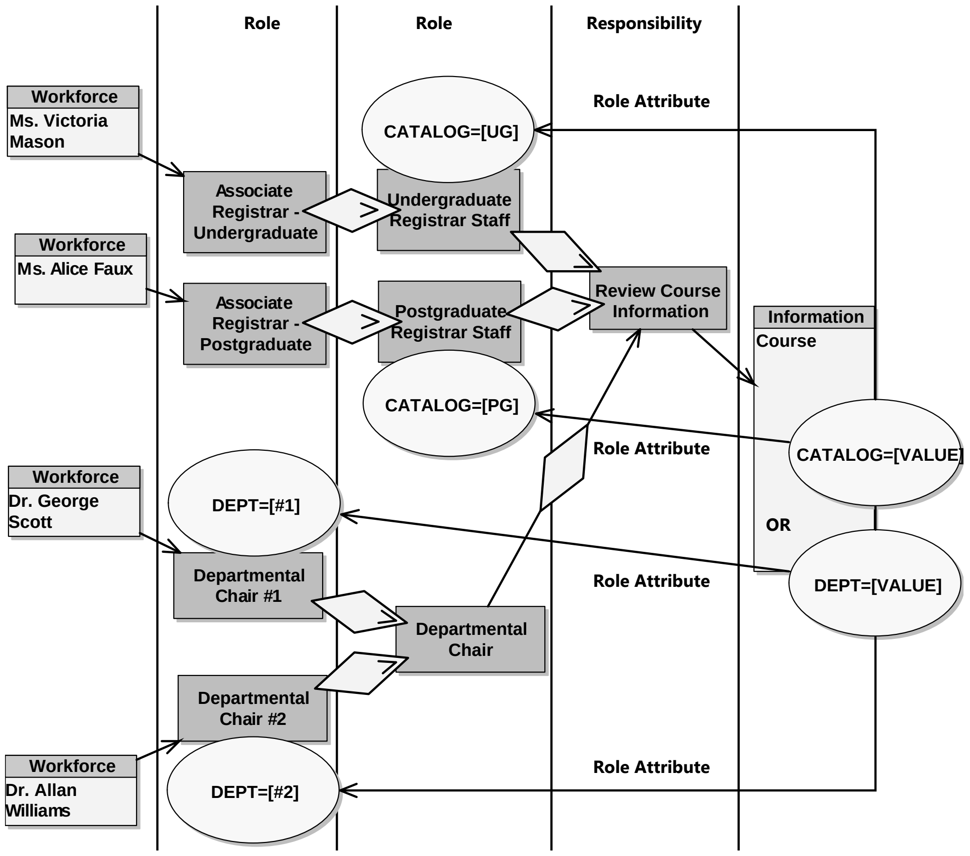

Figure 5.11: Determining the List of Courses that may be selected by a variety of subjects.

- We validate the utility of our new model and methodology, reverse engineering the example from our first operational case study at ACME university
- We discuss how ORGODEX saves time for large organizations with highly diversified workforces

Using real-world data and experiences, we further motivate a mandatory separation between roles and responsibilities, permitting engineers to more easily communicate the important aspects of their design implementations to one another. We stress the importance of communication with the business side of an organization as a primary motivation for ORGODEX, producing a model and methodology where the terms of reference captured by a BA are leveraged by a SA to realize an access control system that is regularly reported in a comprehensible roles and responsibilities matrix, thereby improving the security posture.

We suggest that the time investment for implementing an ORGODEX solution may be measured in minutes and that the expected payoff is the accumulated hours, days and weeks saved when maintaining large-scale RBAC implementations over years and decades. The reader should now better understand why the ORGODEX model and methodology saves time for large organizations with highly diversified workforces, like ACME university, recognizing the added value of working away from the classic triangular model and towards a granular, evolutionary model where roles and responsibilities have semantically diverged to improve communication, thereby lowering the cost of maintenance.

In the following chapter, we further validate the general applicability of ORGODEX, leading a one year project to extend features and functionalities of the ACME university information system to a partner institution.

5.7 Publications

Publications related to this chapter:

- Aaron Elliott and Scott Knight. Start Here: Engineering Scalable Access Control Systems. Proceedings of the 21st ACM Symposium on Access Control Models and Technologies, pages 113-124, 2016.

6 Validation

6.1 Introduction

In the previous chapter we introduced ORGODEX, our new model and methodology for engineering scalable access control systems, building upon our notion of role evolution and providing clear separation for WHO (R^o) and WHY (R^e). In this chapter we validate our new alternative for engineering scalable authorization solutions during a real-world project to deploy both software and authorization services at two geographically distributed partner institutions. To recap, the validation approach for this dissertation is divided into three distinct phases:

- In chapter 3 and chapter 4, we refine our contributions, introducing and motivating the requirement for new RBAC structural relationships that distinguish between roles and responsibilities, thereby satisfying our first objective. We extend RBÄC with role-centric constraints, describing how ACME university decouples subject and permission management to manage role explosion [20].
- In chapter 5 we introduce our new ORGODEX model and methodology for engineering scalable RBAC implementations. We use our first operational case study at ACME university to further motivate the requirement for distinguishing between roles and responsibilities, advocating on behalf of *role evolution* [21].
- In this chapter, our objective is to demonstrate the general applicability of ORGODEX, validating our new model and methodology for engineering scalable authorization solutions in the context of cloud computing. In our second operational case study, we assume the duties of a Project Manager and Security Architect to launch a new shared software solution at two geographically distributed partner institutions where security requirements for data collaboration and isolation are implicit deliverables.

Our objective in this chapter is to demonstrate the efficiency and practicability of our new model and methodology named ORGODEX by leading the deployment of both Software as a Service (SaaS) and Authorization as a Service (AaaS) at two geographically distributed partner institutions. The project team is a multi-disciplinary, multi-lingual group of individuals with extensive to very little understanding of information security and RBAC.

First, during the Analyze phase, the project team collaboratively identifies roles and responsibilities, cohesively aligning role and requirements engineering efforts. We show that ORGODEX is a suitable methodology for aligning project deliverables and authorization from the onset, describing how terms of reference are formalized during early, iterative requirements engineering phases.

Next, during the Realize phase, terms of reference from the Analyze phase, are used to engineer the RBAC implementation. We iteratively design, review, implement and test new features and functionalities before linking roles and responsibilities to realize AaaS. We show that there is a natural progression from the terms of reference distilled during the Analyze phase, to a validated real-world authorization service that is well-understood by the project team.

Finally, during the Publicize phase, we report the realized roles and responsibilities in comprehensible matrices. We demonstrate how ORGODEX supports the ongoing validation of both the software and authorization services, using terms of reference that directly reflect the business model of an organization to facilitate communications.

At the conclusion of this chapter, the reader will better understand why moving towards a roles and responsibilities based RBAC model is an evolutionary approach. We provide evidence for the broader applicability of ORGODEX, further validating this model and methodology in the context of a cloud-computing architecture, satisfying our objective and providing an interesting solution to the challenging problem of entitlements review.

6.2 Background

Since its inception, the cloud-computing paradigm has gained widespread popularity in both industry and academia [1]. With both economical and scalability advantages, shifting business processes to the cloud is a logical evolution for large distributed organizations. However, the cloud requires dynamic, fine-grained access control mechanisms to support employee turnover and adapt quickly to ongoing security policy changes [79].

Unlike other solutions for moving security to the cloud that are based upon different models and technologies such as ABAC and XACML [50] [48], we present a solution that is entirely founded in RBAC. Like the Access Control for Cloud Computing (AC3) model we propose similar principles, enrolling subjects into a role that relates to their job and assigning responsibilities to this role [86].

RBAC has been the dominant access control model for the application layer of computer systems for several decades now [68]. Cloud computing is considered one of the most dominant paradigms in the IT industry today, supporting new cost effective SaaS solutions. Marrying these technologies presents interesting alternatives for partner institutions while presenting unique security challenges that may not be fulfilled by RBAC in a collaborative cloud computing Multi-Tenant Architecture (MTA) [86].

This is an important concern because role engineering is expensive [76][58][1]. Large organizations who have significant investments in RBAC implementations with hundreds of roles engineered over decades to realize security policy and safeguard information assets would like to reuse their implementation when deploying services to partner institutions. Fostering multi-tenant collaboration supported by AaaS is a desirable solution for cloud computing environments where tenants are often isolated, minimizing collaboration [81].

The University Information System Application (UISA) permits users to access information in the IM system of ACME university, supporting multiple higher education institutions, thousands of students, hundreds of staff members and dozens of contractors in an environment operating under a *need to know* security policy. ACME has a significant investment in a large RBAC implementation where hundreds of roles have been engineered over the past two decades to realize security policy and safeguard information assets.

MECA is a partner institution whose campus is located more than three hundred kilometers from ACME university. Students admitted to MECA typically progress in their academic studies and move on to ACME within one to two years, forming an integral connection between the institutions and justifying the requirement for a shared information system. However, failed attempts to meet this requirement historically suggested success would neither be trivial nor a guaranteed outcome for the newly formed project team.

Nevertheless, interest in realizing a shared information system persisted and stakeholders at both institutions prioritized a new project in the spring of 2015, with a Project Director (PD) from ACME working with two BAs from MECA to acquire sponsorship and a commitment to staff the initiative (Table 6.1). In January 2016, the PD assigned one dedicated Developer (DEV) and a part-time Project Manager (PM) to the UISA-MECA-I project. The

Table 6.1: Project Team for UISA-MECA-I including their location, ACME or MECA, and their time commitment to the initiative, part-time (PT) or full-time (FT).

Role	**Responsibilities**
Sponsor	Project Approval and Resources
PT ACME Project Director (PD)	Resources and Reports
PT ACME Project Manager (PM)	Communications and Deliverables
FT ACME Developer (DEV)	Communications and Development
2 PT MECA Business Analysts (BA)	Requirements and Validations

Table 6.2: Project Plan for UISA-MECA-I expected to be a series of one to many projects.

Goal	**Description**	**Milestone**
Communicate	Project initiation	FEB 2016
Integrate	First set of deliverables	MAY 2016
Integrate	Second set of deliverables	AUG 2016
Integrate	Third set of deliverables	DEC 2016
Communicate	Project completion.	JAN 2017

PD's description for the initiative can be paraphrased as follows: *The UISA-MECA-I project will replace MECA's current obsolete IM systems with a more sustainable long term solution, aligning business processes and improving communications between MECA and ACME.*

Coordination is an inherent aspect of work in any organization and takes place in the form of meetings, scheduling, milestones, planning and processes [65]. In early 2016, the newly assigned PM worked with the project team to define scope, milestones and deliverables for the UISA-MECA-I project with the expected business outcomes of reduced administration and improved communication.

The roman numeral notation in the acronym *UISA-MECA-I* indicates the project is expected to be the first in a series of projects delivering UISA functionality to MECA. As listed in Table 6.2, the project scope was confirmed in February 2016 with three sets of deliverables targeted for May 2016, following the winter term, August and December 2016 preceding and following the fall term respectively.

In the following section, we focus on the added benefits of harmonizing role and requirements engineering efforts before designing an access control

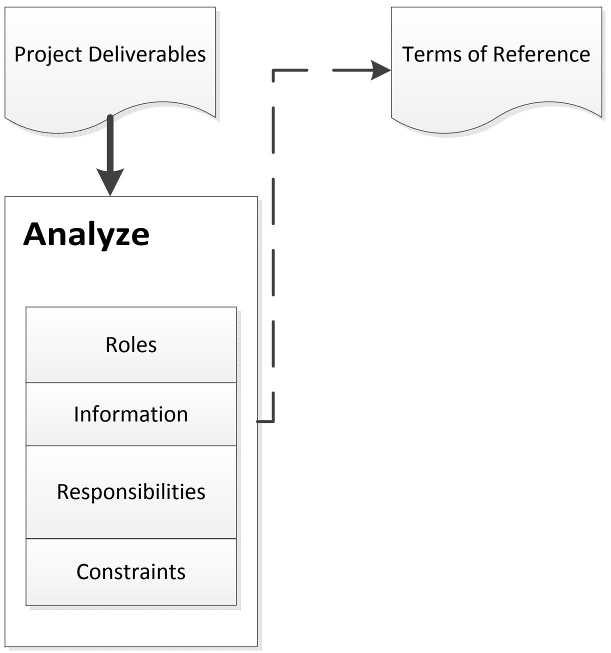

Figure 6.1: During the ORGODEX Analyze Phase, project deliverables serve as inputs and terms of reference including roles, information, responsibilities and constraints are the resulting outputs

model. We show that ORGODEX is a suitable methodology for cohesively aligning project deliverables and access control from the onset, describing how terms of reference are formalized during early, requirements engineering phases. ORGODEX is an iterative process. We presume the reader will deduce that multiple iterations of the analyze, realize and publicize cycles have resulted in the body of the following sections, despite their sequential presentation. What may not be apparent to the reader is the velocity at which these cycles are occurring, often resulting in deployed roles and responsibilities within days of inception. In the real-world case study that follows, we describe our maiden voyage with the ORGODEX model and methodology.

6.3 Analyze

In January 2016, the BAs at MECA produced a 744 word, three page requirements document following interviews with various stakeholder groups at MECA. Features and functionalities were requested by each of the stakeholder groups and captured in bullet form with requirements ranging from partial sentences such as *need to have a system to track contact information* to two word requirements such as *enter grades*.

In February 2016, acting as the PM and SA we used this document as the initial input to the ORGODEX methodology with the belief that we could rapidly realize an access control system, tightly coupled with the project deliverables and well-understood by the team.

The Analyze phase is collaboratively completed by the project team with the PM making requests to a BA using the following prescriptive, mandated steps to analyze project deliverables and produce the following terms of reference:

1. Roles
2. Information
3. Responsibilities
4. Constraints

BAs are expected to elicit outputs by asking questions related to the project deliverables and using the five W's; who, what, why, when and where to ultimately produce a roles and responsibilities matrix.

6.3.1 Roles (R^o)

To build one's instance of ORGODEX, an analysis begins by identifying WHO the information management system and its access control model will support, analyzing the roles to better understand the workforce. This is a critical indicator of scope.

The BAs at MECA estimated an initial workforce of 12 members for the first phase of the envisioned SaaS solution. Subsequently, as mandated by ORGODEX, we requested that the BAs complete a document identifying positions, appointments and groups requiring access. Two days later, the completed document was returned defining 5 positions, 2 appointments and 8 groups with an estimated 37 members of the workforce requiring access, nearly three times greater than the original estimate, providing a better indication of project scope.

It is important for the reader to consider the impact of this straight forward exercise and the rationale for its placement as the lead output of the analyze phase. From the onset, prior to any development activities the project team collaboratively produces a simple document (Table 6.3) identifying the intended workforce (W) and their respective roles (R^o), potentially redefining the scope of the project with the shared realization that the workforce is larger

Table 6.3: Role Definitions for UISA-MECA including the number (#) of persons associated with each role and whether or not these individuals have access to MECA's current system (Yes or No)

Role (R^o)	**Type**	**Description**	**#**	**Access**
P1	Position	Director 1	1	No
P2	Position	Director 2	1	No
P3	Position	Administrator	1	Yes
P4	Position	Director 3	1	No
P5	Position	Admin. Assistant	1	No
A1	Appointment	Dean 1	1	No
A2	Appointment	Dean 2	1	No
G1	Group	Group 1	4	Yes
G2	Group	Group 2	6	Yes
G3	Group	Group 3	3	Yes
G4	Group	Group 4	2	Yes
G5	Group	Group 5	4	Yes
G6	Group	Group 6	2	No
G7	Group	Group 7	2	No
G8	Group	Group 8	7	No
			37	

than initially anticipated. Furthermore, a simple tool for better understanding roles and their maintenance, including addition and subtraction is quickly established by separating WHO (R^o) from WHAT (I) and WHY (R^e) at the onset.

6.3.2 Information (I)

Next, as mandated by ORGODEX, we used the requirements document to abstract informational concepts such as Student, Person and Applicant, documenting and further clarifying our terms of reference (WHAT) while simultaneously identifying the corresponding protection levels.

In Table 6.4 we produce a small sampling of our information analysis document. The rationale for determining the protection levels early in the analysis process is twofold. First, we must establish a balance between security and practicality from the onset. Second, we want to confirm protection levels for information before performing a responsibility analysis.

Table 6.4: An Extract of the Information Assets defined for the UISA-MECA project including whether or not the item is deemed Protected or Public (Yes or No)

Information (I)	**Description**	**Protected**
Person	Consolidates information for individuals with multiple profiles. At a higher learning institution, an individual can be a Staff member and a Student, for example	Yes
Applicant	Relates information concerning an individual applying for admission	Yes
Student	Identifies individuals engaged in the pursuit of higher education	Yes
Course	Identifies the subject and learning objectives to be delivered in a defined period of time (e.g. English 101)	No
Program	Defines the cumulative learning competencies that must be achieved to be awarded certification (e.g. Undergraduate Degree)	No

If an information set is deemed *PROTECTED* an organization must apply safeguards to ensure its safe use and distribution. The *need to know* principle or policy of least privilege proposes that access to sensitive information must be limited to those whose responsibilities require such access.

6.3.3 Responsibilities (R^e)

ORGODEX prescribes the identification of responsibilities, intentionally separating WHO (R^o) and WHY (R^e) from the onset to facilitate communication amongst project team members early and often, working towards a shared understanding of the access control system-to-be and prioritizing requirements, ensuring that they can be read, analyzed, (re-)written, and validated [56].

The project team collectively distilled requested features and functionalities into high-level responsibilities, using a highly iterative process where Table 6.5 lists only a few items of interest from the responsibility analysis document.

Consider the impact of this ORGODEX work product and its timing. From the onset, prior to any development activities the project team collaboratively

Table 6.5: A sampling of the responsibility analysis document for UISA-MECA-I including the related information (I) concept and a defined responsibility (R^e).

Information (I)	**(R^e)**	**Responsibility**
Student	R1	View List of Active Students
Student	R2	Maintain List of Active Students
Person	R3	Maintain Personnel File
Person	R4	View List of Active Personnel
Person	R5	Maintain List of Active Personnel
Applicant	R6	View List of Applicants

Table 6.6: The Appendix to the Project Plan for UISA-MECA-I defines deliverables (or responsibilities) to be implemented along with the target milestone.

Deliverable	**Milestone**
View List of Active Students	2016-05
Maintain List of Active Students	2016-05
Maintain Personnel Files	2016-05
View List of Active Personnel	2016-05
Maintain List of Active Personnel	2016-05
View List of Applicants	2016-05
View Language Results	2016-08
Maintain Language Results	2016-08
Evaluate List of Applicants	2016-08
View Athletic Results	2016-08
Maintain Athletic Results	2016-08
Support Multiple Languages	2016-08
Maintain Student File	2016-12
View Academic Results	2016-12
Maintain Academic Results	2016-12

produces a simple document (Table 6.5), identifying the Information (I) assets and the related responsibilities (R^e), collaboratively working towards a shared understanding of the security architecture. Furthermore, the maintenance, including addition, modification and subtraction of responsibilities is established in a very short period of time by determining WHY (R^e) access to information is required.

As stated earlier, communication is often considered the most important aspect of Project Management, justifying the need to work towards shared goals, objectives and outcomes that must be communicated out from the project team to stakeholders (Table 6.6). With this objective in mind, the PM coordinated a teleconference with the BAs to outline the purpose of a responsibility analysis document and propose next steps. The BAs agreed to confirm priorities with stakeholders at MECA. By determining WHY (R^e) access to information is required with the understanding that WHO (R^o) requires access is subject to change in both the short and long term.

6.3.4 Constraints (C)

Finally, ORGODEX mandates a constraints analysis before linking roles and responsibilities, formalizing WHEN (C) and WHERE (C) access to information is required. As one example, it was generally acknowledged by the project team and expected that the workforce at MECA would only have access to students at their institution. In the SaaS model proposed for the project, all active students at both ACME and MECA were listed in the same logical database table, making it necessary to define constraints (C) when accessing sets of active students.

The result of a constraints analysis exercise is a roles and responsibilities matrix, identifying WHO - HAS ACCESS TO - WHAT, indicating WHY, WHEN and WHERE as applicable. In Table 6.7, we list the constraints placed upon all groups identified in the role analysis document when viewing the list of active students, documenting the requirement for all groups at MECA $G_1 ... G_m$ to access the functionality to *View List of Active Students*. We also transparently acknowledge the parallel intent to deploy the SaaS solution at ACME, indicating that ACME groups $G_i ... G_n$ may one day share the system-to-be and indicating this requirement is out of scope for UISA-MECA-I**.

Although we were confidently informed all members of the workforce at MECA, with access to UISA, would have the responsibility *View List of Active Students*, we also noted that student information was previously labeled protected, making us wary of this particular authorization. With the ORGODEX methodology we anticipate change, singularly defining the responsibility to

Table 6.7: A constraints analysis document identifies WHO - HAS ACCESS TO - WHAT, indicating WHY, WHEN and WHERE as applicable. **ACME requirements are transparently acknowledged but not in scope for the UISA-MECA-I project.

WHO (R^o)	**WHAT (I)**	**WHY (R^e)**	**WHEN (C)**	**WHERE (C)**
$G_1 ... G_m$	Student	View List of Active Students	Student is administered by MECA	UIS
ACME** $G_i ... G_n$	Student	View List of Active Students	Student is administered by ACME or MECA	UIS

View List of Active Students. This is an important design consideration as we will see in the following section as the requirement for all personnel at MECA to access the functionality *View List of Active Students* had the distinct odor of a short-sighted assumption.

A secondary longer term objective not scoped for the UISA-MECA-I project was well-understood by the PM. The old desktop version of UISA, a custom in-house software developed by the IM team for the Microsoft^{TM}Windows operating system and acting as a client to the UIS database for ACME personnel had lagged well behind more current technologies, making its replacement a desirable outcome running parallel to the UISA-MECA-I project.

In this section, we empirically verify the efficiency and practicability of the ORGODEX model and methodology, demonstrating how project deliverables are used to directly influence the RBAC implementation, thereby confirming our belief that project management and access control can be seamlessly integrated from the onset. Throughout this section we list work products delivered by a geographically distributed project team whose members have extensive to very little understanding of RBAC and security requirements.

In this fashion, we risk trivializing the important tangible results hiding in the details. The assumption being that the reader will understand that it is the project team performing each and every phase of the methodology to analyze project deliverables, collaboratively define terms of reference and realize an access control system that is well-understood by all members of the

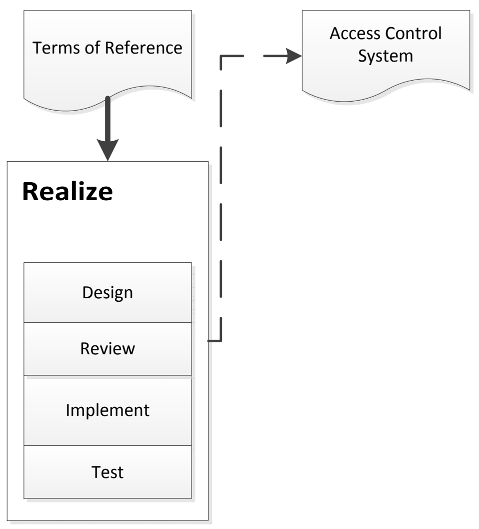

Figure 6.2: During the ORGODEX Realize Phase, terms of reference identified during the Analyze Phase, including roles, information, responsibilities and constraints serve as inputs to produce an access control system as the output.

project team, facilitating its ongoing maintenance and scalability.

In the following section, we describe how we directly leverage the project team's work products during the realize phase of ORGODEX. We presume the reader will infer that we have iteratively designed, reviewed, implemented and tested new features and functionalities before linking roles and responsibilities to realize an access control system.

6.4 Realize

In February 2016, acting as both the PM and SA for the project, we initiated the design, review and implementation of our authorization service. In the following subsections we demonstrate how the outputs of the ORGODEX Analyze phase serve as inputs to the Realize phase. We show that there is a natural progression from the terms of reference (R^o, I, R^e, C) to a validated real-world access control system that is well-understood by the project team

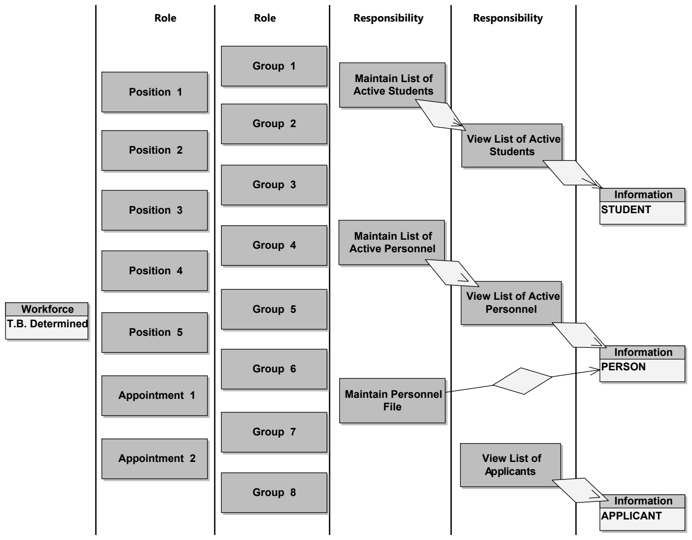

Figure 6.3: During the design phase we propose responsibility hierarchies (R^eH) and identify roles (R^o) typed as positions, appointments and groups.

and communicated out to the business.

6.4.1 Design

During the Design phase we propose hierarchies for responsibilities (R^eH), producing diagrams to confirm our understanding with the BAs. Malone and Crowston emphasize that software architecture can serve as a coordination tool where distributed project teams must collaborate with visual aids to develop a shared understanding of the system-to-be [53].

6.4.2 Review

During the Review phase, we utilize ORGODEX hierarchical diagrams to better visualize the directional hierarchy of workforce-role, role-responsibility and responsibility-information grants to confirm our design [20]. In Figure 6.3, we include our proposed responsibility hierarchies, visually depicting dependencies between responsibilities. We also identify roles typed as positions, appointments and groups to help the project team visualize the logical design.

With independent roles and responsibilities, we initiate a simple framework for developers to focus on the functional requirements, assigning permissions to read and write data to responsibilities from the onset as we describe in the following subsection.

6.4.3 Implement

During the Implement phase of ORGODEX, a reviewed list of roles (R^o) as defined in Table 6.3 enables us to issue the CREATE ROLE command and implement 5 positions, 2 appointments and 8 groups as follows:

- CREATE R^o *P1 ... P5*;
- CREATE R^o *A1 ... A2*;
- CREATE R^o *G1 ... G8*;

Next, with a reviewed list of responsibilities (R^e) as defined in Table 6.5, we issue the CREATE RESPONSIBILITY command to implement 6 responsibilities:

- CREATE R^e *R1 ... R6*;

Then following our design reviews, we implement two responsibility hierarchies (R^eH), issuing the following commands:

- CREATE R^eH between R1 and R2;
- CREATE R^eH betwee R4 and R5;

Finally, we must add constraints. In the following example, we add ORGODEX role-centric constraints permitting the group roles G_1 to G_m to view the list of active students (R1) administered by MECA:

- ADD ATTRIBUTE A_1 = (ADMIN, MECA)
- ADD CONSTRAINT (G_1, R1, A_1)
- ADD CONSTRAINT (G_m, R1, A_1)

Constraints are applied by creating database views with support for ORGODEX role-centric constraints. For instance, MECA is substituted for [value] in the following database view when a member of G_1 is viewing the list of active students.

- SELECT information FROM student WHERE admin IN [value]

It is important for the reader to note that unless constraints are defined, no records are returned for groups accessing this database view, imposing a strong default security posture. For the ACME group roles G_i to G_n multiple constraints are defined, permitting access to view the active list of students at both partner institutions:

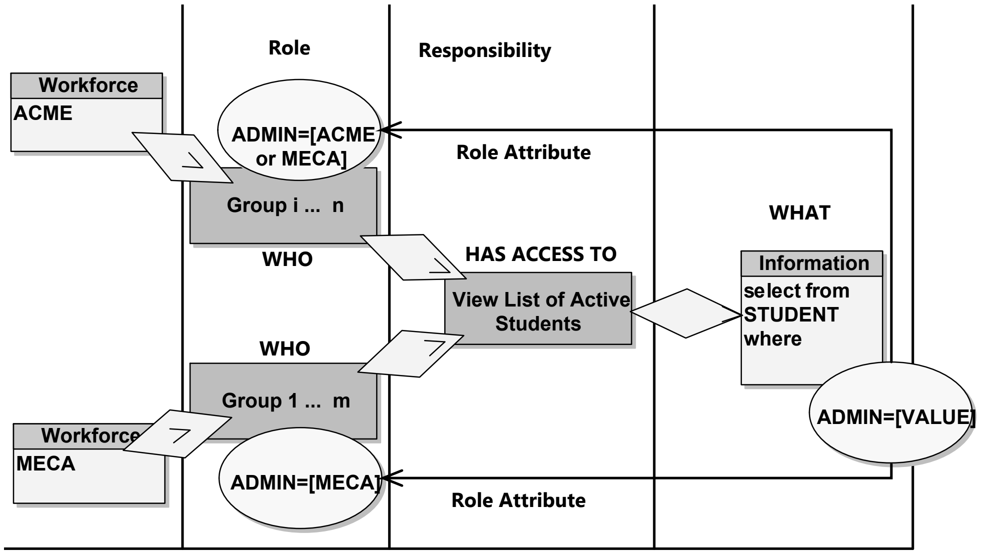

Figure 6.4: During the Test Phase, we expect different results due to the constraints applied to various ACME and MECA groups.

- ADD ATTRIBUTE A_2 = (ADMIN,ACME)
- ADD CONSTRAINT (G_i, R1, A_1)
- ADD CONSTRAINT (G_i, R1, A_2)
- ADD CONSTRAINT (G_n, R1, A_1)
- ADD CONSTRAINT (G_n, R1, A_2)

6.4.4 Test

During the Test phase of ORGODEX, developers ensure that features meet the constraints identified during the analyze phase prior to coordinating a review and approval demonstration. For instance, the project team expects staff members at MECA to have access to students at their institution only (Figure 6.4).

At review and approval demonstrations with the project team and relevant stakeholders, feedback obtained is used to complete the remaining work before granting responsibilities to roles in the production RBAC implementation, for example:

- GRANT R1 to G5;
- GRANT R3 to G5;

- GRANT R6 to G5;

Although we were confidently informed all individuals at MECA, would need the *View List of Active Students* responsibility (R1), in October 2016, we were informed Groups 4 and 5 no longer required this access. With the ORGODEX methodology we anticipate change, and defining the responsibility to *View List of Active Students* makes this change trivial:

- REVOKE R1 from G4;
- REVOKE R1 from G5;

6.5 Publicize

In the Publicize phase, we report the realized roles and responsibilities of our AaaS service directly within the deployed SaaS solution, facilitating its ongoing validation with terms of reference that directly reflect the business model of MECA. The intent is to display real security architecture information on demand for authorized members of the workforce using a comprehensible roles and responsibilities matrix to regularly validate profiles.

6.5.1 Information

Over time a business develops and operates using a unique glossary or terms of reference. During the UISA-MECA-I project well-understood concepts like student and course were defined in addition to very context specific terms of reference such as UISA. For this reason, it is important for large organizations to develop and maintain searchable terms of reference.

The Information Asset Report (Table 6.8) defines terms of reference that are used by BAs, SAs and Developers. These terms are also communicated out to the organization directly within the deployed SaaS solution as a searchable glossary that may be regularly validated as part of the ORGODEX publicize cycle.

6.5.2 Roles and Responsibilities

Ensuring that the workforce has access to the required information is a challenging problem for professionals [6]. RBAC implementation changes are required when new hires enter the organization, former employees leave and individuals move from one position to another. RBAC implementations must adapt to new security policies and new software applications. Using the terms

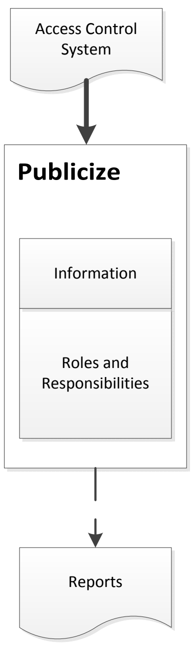

Figure 6.5: During the ORGODEX Publicize phase, the access control system serves as the input and reports related to information assets, roles and responsibilities are the output.

role and responsibility is a practical approach to implementing RBAC, permitting BAs, SAs and Developers to more easily communicate the important aspects of their design implementations to one another and the business.

The Roles and Responsibilities Report in particular is an interesting solution to the challenging problem of entitlements review [38]. We have observed the utility of this report in practice for more than twelve months at MECA where it has proven to be a valuable communication tool. With ORGODEX the administration of roles and responsibilities is just another responsibility, thereby providing cost effective maintenance without sacrificing efficiency [43]. Table 6.9 identifies WHO - HAS ACCESS TO - WHAT, indicating WHY, WHEN and WHERE.

Table 6.8: The Information Asset Report defined for the UISA-MECA project including whether or not the item is deemed Protected or Public (Yes or No)

Information (I)	**Description**	**Protected**
Person	Consolidates information for individuals with multiple profiles. At a higher learning institution, an individual can be a Staff member and a Student, for example	Yes
Applicant	Relates information concerning an individual applying for admission	Yes
Student	Identifies individuals engaged in the pursuit of higher education	Yes
Course	Identifies the subject and learning objectives to be delivered in a defined period of time (e.g. English 101)	No
Program	Defines the cumulative learning competencies that must be achieved to be awarded certification (e.g. Undergraduate Degree)	No

Table 6.9: The Roles and Responsibilities Report identifies WHO - HAS ACCESS TO - WHAT, indicating WHY, WHEN and WHERE as applicable.

WHO R^o	**WHAT (I)**	**WHY (R^e)**	**WHEN (C)**	**WHERE (C)**
Group 4	Student	View List of Active Students	Student is administered by MECA	UIS
Group 4	Student	Maintain List of Active Students	Student is administered by MECA	UIS
Group 4	Person	Maintain Personnel File	Person is a Student administered by MECA	UIS
Group 4	Applicant	View List of Applicants	All Applicants administered by MECA and ACME	UIS
Group 4	Applicant	Evaluate List of Applicants	Applicants administered by MECA	UIS

6.6 Discussion

In this chapter, we detail aspects of a project to extend features and functionalities of a university information system to a geographically distributed partner institution. The empirical study described provides architectural information for an access control system developed and deployed on an Oracle^{TM}technology stack in April 2016. Both our SaaS and AaaS solutions have scaled to support many more roles and responsibilities since its initial deployment more than twelve months ago.

Unlike other models found in the literature, where validations are limited to prototypes or example policy specifications [84] [81], this is a real-world RBAC implementation. Moreover, unlike other authors, we explain that large organizations who have RBAC implementations with hundreds of roles engineered over decades to realize security policy would like to reuse their implementation when deploying services to distributed partner institutions.

In this work, we do exactly that, extending the RBAC implementation of ACME university to MECA, deploying AaaS and empirically verifying the efficiency and practicability of the ORGODEX model and methodology, thereby offering a new integrated alternative for the most expensive aspect of deploying RBAC [58]. We cohesively align role and requirements engineering efforts, validating the work of Coyne who suggests that the definition of roles is essentially a requirements engineering process [13].

First, we assume the duties of both a PM and SA to analyze project deliverables with our team, demonstrating how they are used to directly influence the RBAC implementation and confirming our belief that project management and access control can be seamlessly integrated from the onset. We show that our terms of reference (R^o, I, R^e, C) are critical indicators of project scope and we produce simple communication tools to clearly separate WHO (R^o) from WHAT (I) and WHY (R^e) at the onset before discussing constraints such as WHEN (C) and WHERE (C).

Next, we realize our first instance of an ORGODEX AaaS that is tightly coupled with our terms of reference. Then we deploy the first version of our SaaS solution to MECA less than three months after project initiation, utilizing our new authorization service to deliver information on a *need to know* basis. It is important for the reader to understand that both our AaaS and SaaS solutions are collaboratively developed and validated by a multi-disciplinary, multi-lingual project team with extensive to very little understanding of information security and RBAC.

Finally, we explain why the Role and Responsibilities Report is a practical approach to implementing RBAC, permitting the PM, BAs, SAs and

Developers to more easily communicate the important aspects of the security architecture to one another and the business, thereby facilitating the scalability and on-going maintenance of an ORGODEX AaaS implementation.

Although the cloud-computing environment is the platform described in this chapter, the reader should understand that ORGODEX is a generalized solution for engineering scalable RBAC implementations on a variety of platforms, including but not limited to relational database management systems and cloud computing environments. During our validation, we often considered the notion of a shared RBAC repository where practitioners could collaboratively design and review domain specific RBAC implementations.

ORGODEX proposes a model and methodology for delivering this platform, permitting engineers to more easily communicate the important aspects of their designs to one another. We consider the notion of shared critically reviewed RBAC implementations to be a new paradigm for RBAC. To the best of our knowledge, this is a new proposition for RBAC, an open, generic technology where, for most practitioners, information security is a secondary duty.

6.7 Summary

This research has the potential to impact the perception of access control, providing consistent role and responsibility based security architectures that are subject to critical analysis and reuse. In this chapter, we make the following contributions:

- We empirically verify the efficiency and practicability of ORGODEX in the context of a cloud-computing MTA, demonstrating how project deliverables are used to directly influence a realized and publicized real-world RBAC implementation.
- We validate a new RBAC alternative for large organizations who do not see value in redesigning their information security architecture to use the ABAC model and its XACML policies.
- We show how communication is facilitated, listing work products collaboratively delivered by a project team whose members have extensive to very little understanding of RBAC and information security.

The reader should now better understand why evolving towards a role and responsibility based RBAC model is a suitable progression for AaaS solutions, enabling distributed project teams with multi-disciplinary skill sets to collaboratively develop and maintain a shared understanding of the RBAC implementation, thereby facilitating its ongoing validation.

6.8 Publications

Publications related to this chapter:

- Aaron Elliott and Scott Knight. ORGODEX: Authorization as a Service. Proceedings of the 12th Annual IEEE International Systems Conference, to appear 2018.

7 Conclusion

7.1 Introduction

In the previous chapter we perform validation activities for ORGODEX, our new model and methodology for engineering scalable access control systems, building upon our notion of role evolution. In this chapter, we conclude this dissertation, beginning with a final summarization of the validation approach:

- In chapter 3 and chapter 4, we refine our contributions, challenging the belief, notion or sense that the number of subjects far exceeds the roles found in enterprise systems, thereby motivating a return to first principles. Next, we compare the information security of ACME university to bi-sorted role-based access control (RBÄC), using real-world data and experiences to support this new principled approach to RBAC. Then, we extend RBÄC with role-centric constraints, describing how ACME university decouples subject and permission management at the expense of role evolution.
- In chapter 5 we introduce our new ORGODEX model and methodology for engineering scalable RBAC implementations. We use our first operational case study at ACME university to motivate the requirement for new RBAC structural relationships, distinguishing between roles and responsibilities, advocating on behalf of role evolution. We discuss time savings that may be measured in hours, days and weeks when maintaining RBAC implementations, especially those whose lifetime spans years and crosses decades.
- In chapter 6, we demonstrate the general applicability of ORGODEX, validating this new model and methodology for engineering scalable authorization solutions in the context of cloud computing. In this real-world case study, we assume the duties of a Project Manager and Security Architect to lead a one year initiative to deploy both software and authorization services at two geographically distributed partner institutions. The project team is a multi-disciplinary, multi-lingual group

of individuals with extensive to very little understanding of information security and RBAC.

In the following sections, we begin with a review of our statement of deficiency, reminding the reader of the relevance of this work. Next, we restate our aim, identifying our objectives before describing how they have been met. Then we elaborate upon the validations completed, specifically making our arguments for sufficiency, validity and feasibility. Finally, we list our contributions and suggest streams for future work.

At the conclusion of this chapter, the reader will better understand why evolving towards a roles and responsibilities model like ORGODEX is a suitable progression for RBAC, improving communication activities and introducing time savings that may be measured in hours, days or weeks.

7.2 Deficiency

Although RBAC is a popular solution for implementing information security our research suggests there is no pervasive methodology used by practitioners when producing scalable access control systems for large organizations with hundreds or thousands of employees. As a result, there is both an immediate and long term cost related to the difficulty of communicating the important aspects of the RBAC implementation to those responsible for its maintenance. This is an interesting deficiency because despite their diversity, large organizations are built upon two key concepts, roles and responsibilities, where a role like Departmental Chair is identified and assigned responsibilities

Access control is often considered a necessary burden and too often it is an afterthought [17][32][33]. This is disappointing because RBAC has the potential to be a well understood, enabling technology, where implementations directly reflect the business model of an organization. Instead RBAC systems are often loosely coupled with the business model, ripe with redundancies and costly to maintain [12] [52] [58].

Employees are dissatisfied because it is often unclear how to obtain the information they need to perform their job and SAs are overburdened because they are responsible for thousands of fragile subject-role-permission interconnections that are subject to continuous change.

This is an important concern because role engineering is expensive [76][58][1]. Large organizations who have significant investments in RBAC implementations with hundreds of roles engineered over decades to realize security policy and safeguard information assets would like to reuse their implementation when deploying new services and adapting to meet the demands and expectations of clients, stakeholders, partners and employees. For large, mature organizations with decades of IT investments, the challenge of replacing legacy solutions with modern alternatives are well-documented and costly [77][70][70]. IT supports IM which must support the business [82].

7.3 Aim

From section 1.6, the aim or our research is:

To deliver a new model and methodology for engineering scalable access control systems in organizations where hundreds or thousands of employees require access to information on a need to know basis in order to perform their job.

In this dissertation, we refined our contributions, introducing and motivating the requirement for new RBAC structural relationships that distinguish between roles and responsibilities. Then we delivered ORGODEX, our new model and methodology for engineering scalable RBAC implementations, using our operational case studies at ACME university and its partner institution, MECA, to further validate the general applicability of this work.

Our literature review and validation approach identify a deficiency and our research delivers valuable contributions to the field of computer science and information security.

7.4 Validation

In this dissertation it is incumbent that we demonstrate why ORGODEX is valid, successfully providing argumentation for its sufficiency and feasibility in relation to our statement of deficiency and research aim.

7.4.1 Sufficiency

The ORGODEX model and methodology delivers a novel solution to the research problem. The model introduces a new mandatory layer of abstraction, termed a responsibility, to better align with the business model of large organizations, like ACME university. The methodology delivers a highly iterative, repeatable framework for engineering scalable access control systems based upon comprehensible role and responsibility matrices.

7.4.2 Validity

The success of the RBAC model is due in large part to the administrative savings achieved by creating logical roles versus simply assigning permissions directly to the user. It is easy to demonstrate how trivial time investments result in administrative savings. Likewise, ORGODEX and role evolution introduce trivial time investments that have the potential to save hours, days and weeks.

To successfully argue that the aim of the research has been met, we must demonstrate how this dissertation addresses our statement of deficiency. To this end our argumentation may be best summarized as follows:

- If role explosion is a normal occurrence and previous models for the administration of RBAC consider role explosion a design problem, this should motivate the requirement for role evolution and the introduction of new RBAC structural relationships that distinguish between roles and responsibilities.
- Role evolution does not aim to prevent role explosion, instead it presumes the number of roles will scale uncontrollably in large organizations or controllably using a prescribed methodology.
- ORGODEX is a new granular, evolutionary model and methodology for describing and reasoning about RBAC implementations, using roles and responsibilities to directly inform the architecture.
- The general applicability of ORGODEX has been demonstrated using two operational case studies at ACME university and its partner institution, MECA. We have demonstrated the utility of our new model and methodology, observing improved communications when reasoning about RBAC structural relationships and using the terms role and responsibility to better align with the business model.

7.4.3 Feasibility

We demonstrate the feasibility of ORGODEX with each of our operational case studies:

- We validate the utility of our methodology by reverse engineering our ACME university scenario of interest where each Departmental Chair (DC) has similar responsibilities but different sets of students, courses and programs to manage. We demonstrate how role-centric constraints are applied to responsibilities to restrict access to information.

- We further validate the general applicability of our new model and methodology during our second operational case study, extending features and functionalities of the ACME university information system to it partner institution, MECA. We use ORGODEX to deploy software and authorization services in April 2016, realizing many more roles and responsibilities since the initial deployment more than twelve months ago.
- The reader should now better understand why ORGODEX is a suitable progression for RBAC, enabling project teams with multi-disciplinary skill sets to collaboratively develop and maintain a shared understanding of both roles and responsibilities, thereby facilitating the ongoing validation of an RBAC implementation.

7.5 Contributions

This research makes several contributions to the field of computer science and information security:

- Our most important contribution is the delivery of our new methodology for engineering scalable RBAC implementations that are well understood, directly reflect the business model and facilitate communications.
- Our ORGODEX model introduces a new layer of abstraction for RBAC implementations, where both roles and mandatory responsibilities are created, permitting engineers to more easily communicate the important aspects of their design implementations to one another.
- Our description of role evolution is a novel concept forming the foundation of our ORGODEX model. We propose a mandatory divergence between roles and responsibilities.
- Role evolution is based upon a return to first principles. We extend the work of Kuijper and Ermolaev [46], adding role-centric constraints to their model and introducing business terminology to facilitate comprehension amongst practitioners.
- We introduce a novel hierarchical diagramming notation to better visualize explicit RBAC implementations. We show why implicit subject-permission relationships hide too many details, trivialize RBAC implementations and impede communication.
- We dispel the belief, notion or sense that some threshold ratio of roles to subjects indicates bad RBAC design. We discuss why role explosion naturally occurs in large organizations with highly diversified workforces.

- We empirically verify the efficiency and practicability of ORGODEX, demonstrating how project deliverables are used to directly influence a realized and publicized real-world RBAC implementation.
- We have already contributed to the field of research through publication in [18][19][20][21][22]. These works acknowledge role explosion, express ideas aimed towards role evolution and propose our ORGODEX model and methodology for engineering scalable access control systems.

7.6 Future Work

There are many streams for future work that we simply did not have time to see through to fruition prior to the completion of this dissertation including the following:

- We have developed a novel database integration component for applying our ORGODEX model and methodology to both new and existing information systems. We are continuing our development of the supporting software and we look forward to one day publishing our results. Although this validation aspect has not been described throughout this dissertation, the reader may have presumed such a tool was under development. We confirm that presumption here but have decided against its inclusion at this time.
- In the chapter 5 discussion we refer to the *Degree of Delegation and Automation* as metrics of interest. In future iterations of the ORGODEX model and methodology these will be important points of reference used to further strengthen the validity of this work. With the ORGODEX model, there is a clear path towards delegation for managers, using responsibilities as the currency. Similarly, ORGODEX demonstrates how a granular, evolutionary model might facilitate the enrollment of employees into a role. Our scenario of interest from ACME university exemplifies a case where the appointment of a Departmental Chair in the HRMS automates the assignment of a role to the individual appointed, for example DC#1.
- The chapter 6 discussion includes insight obtained during our project to extend features and functionality from ACME university to MECA. We consider work directed towards creating and sharing critically reviewed, domain specific RBAC implementations a new paradigm for RBAC. ORGODEX proposes a model and methodology for delivering this platform, permitting engineers to more easily communicate the important aspects of their designs to one another. RBAC is challenging to

implement and even more costly to maintain for large organizations like universities, banks and hospitals.

- In this work we include support for constraining responsibilities. Our operational case study at ACME university describes a scenario where the responsibility to *Review Course Information* is constrained to Dept#1 for DC#1. At ACME university there are also temporal constraints for this responsibility, currently performed twice per academic year following the Fall and Winter sessions. Although we firmly believe the current version of the ORGODEX model supports temporal constraints it would be interesting to validate this belief using a real-world case study.
- Likewise, support for other constraints is of interest. In particular, Static Separation of Duty (SSD) defining mutually exclusive responsibilities and Dynamic Separation of Duty (DSD) identifying scenarios where the same individual may not perform both the initiating and approving responsibility within the context of one process. For SSD the classic example defines a *static* rule where no individuals responsible for initiating payments may perform approvals. For DSD, the classic example also revolves around initiating and approving payments. This constraint is considered *dynamic* because an individual who is responsible for initiating payments may also perform approvals as long as they are not the initiator of the process. Again, we believe that the current version of the ORGODEX model supports both SSD and DSD but would like to confirm this belief with a real-world case study.

7.7 Summary

In this chapter, we conclude this dissertation:

- We review our statement of deficiency and restate our aim before listing our objectives and describing how they are met
- We discuss the validation activities performed in this dissertation and make our arguments for sufficiency, validity and feasibility
- We list future work that we intend to pursue, further strengthening and adding value to our new paradigm for RBAC

The reader should now better understand that role explosion is a normal occurrence for large organizations with highly diversified workforces. Role evolution presumes the number of roles will scale controllably using a prescribed model and methodology. The general applicability of ORGODEX has been demonstrated using two operational case studies where roles and responsibilities have semantically diverged to directly inform the RBAC implementation, improve communications and lower the cost of maintenance.

Bibliography

[1] Mazhar Ali, Samee U. Khan, and Athanasios V. Vasilakos. Security in cloud computing: Opportunities and challenges. *Information Sciences*, 305:357–383, 2015.

[2] American National Standards Institute. INCITS 359-2004: Information Technology - Role Based Access Control, 2004.

[3] American National Standards Institute. INCITS 359-2012: Information Technology - Role Based Access Control, 2012.

[4] James P Anderson. Computer Security Technology Planning Study. Technical report, DTIC Document, oct 1972.

[5] Steffen Bartsch and M Angela Sasse. How Users Bypass Access Control - And Why: The Impact Of Authorization Problems On Individuals And The Organization. In *Proceedings of the 21st European Conference on Information Systems*. Association for Information Systems, 2012.

[6] Lujo Bauer, Lorrie Faith Cranor, Robert W. Reeder, Michael K. Reiter, and Kami Vaniea. Real life challenges in access-control management. *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*, page 899, 2009.

[7] Matthias Beckerle and Leonardo a Martucci. Formal Definitions for Usable Access Control Rule Sets from Goals to Metrics. *SOUPS '13: Proceedings of the Ninth Symposium on Usable Privacy and Security*, pages 2:1—-2:11, 2013.

[8] Iris Berdrow. King among kings: Understanding the role and responsibilities of the department chair in higher education. *Educational Management Administration & Leadership*, 38(4):499–514, 2010.

[9] Konstantin Beznosov, PG Inglesant, Jorge Lobo, R Reeder, and ME Zurko. Usability meets access control: challenges and research opportunities. pages 3–4, 2009.

[10] Matt Bishop, Sophie Engle, Sean Peisert, Sean Whalen, and Carrie Gates. We have met the enemy and he is us. In *Proceedings of the 2008 workshop on New security paradigms - NSPW '08*, volume 15, page 1, New York, New York, USA, 2008. ACM Press.

[11] Eric Chen, Yutong Pei, Shuo Chen, Yuan Tian, Robert Kotcher, and Patrick Tague. OAuth Demystified for Mobile Application Developers. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, (1):892–903, 2014.

[12] Alessandro Colantonio, Roberto Di Pietro, and Alberto Ocello. A cost-driven approach to role engineering. *Proceedings of the 2008 ACM symposium on Applied computing - SAC '08*, page 2129, 2008.

[13] Edward J Coyne. Role engineering. *Proceedings of the first ACM Workshop on Rolebased access control RBAC 95*, (4):4–es, 1996.

[14] Edward J. Coyne and Timothy R. Weil. ABAC and RBAC: Scalable, flexible, and auditable access management. *IT Professional*, 15(3):14–16, 2013.

[15] J Crampton. Understanding and developing role-based administrative models. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 158–167, New York, New York, USA, 2005. ACM Press.

[16] Jason Crampton and George Loizou. Administrative scope. *ACM Transactions on Information and System Security*, 6(2):201–231, may 2003.

[17] Robert Crook, Darrel Ince, and Bashar Nuseibeh. Modelling access policies using roles in requirements engineering. *Information and Software Technology*, 45(14):979–991, nov 2003.

[18] Aaron Elliott and Scott Knight. One Employee and Several Applications: An Information Management Case Study. In *Software Engineering Research and Practice, WORLDCOMP*, pages 179–185. CSREA Press, 2009.

[19] Aaron Elliott and Scott Knight. Role Explosion: Acknowledging the Problem. In *Software Engineering Research and Practice, WORLDCOMP*, pages 349–355. CSREA Press, 2010.

[20] Aaron Elliott and Scott Knight. Towards Managed Role Explosion. In *Proceedings of the 2015 New Security Paradigms Workshop (NSPW)*, number 1, pages 100–111. ACM Press, 2015.

[21] Aaron Elliott and Scott Knight. Start Here: Engineering Scalable Access Control Systems. In *Proceedings of the 21st ACM Symposium on Access Control Models and Technologies*, pages 113–124. ACM Press, 2016.

[22] Aaron Elliott and Scott Knight. ORGODEX: Authorization as a Service. In *Proceedings of the 12th Annual IEEE International Systems Conference*, 2018.

[23] EmpowerID. Best Practices in Enterprise Authorization : The RBAC/ABAC Hybrid Approach. Technical report, EmpowerID, 2013.

[24] Christophe Feltus. *Aligning Access Rights to Governance Needs with the Responsibility MetaModel (ReMMo) in the Frame of Enterprise Architecture.* 2014.

[25] Christophe Feltus, Michaël Petit, and Morris Sloman. Enhancement of Business IT Alignment by Including Responsibility Components in RBAC. *Proceedings of the CAiSE 2010 Workshop Business/IT Alignment and Interoperability*, pages 61–75, 2010.

[26] D F Ferraiolo and R Kuhn. Role-based access controls. *Proc. of 15th NIST-NSA National Computer Security Conference*, 1992.

[27] Philip W.L. Fong. Relationship-based access control. *Proceedings of the first ACM conference on Data and application security and privacy - CODASPY '11*, page 191, 2011.

[28] C Gates. Access control requirements for Web 2.0 Security and Privacy. In *IEEE Web*, volume 2, pages 2–4, 2007.

[29] Luigi Giuri and Pietro Iglio. Role templates for content-based access control. *Proceedings of the second ACM workshop on Role-based access control - RBAC '97*, pages 153–159, 1997.

[30] Michael Goold, Ashridge Strategic, Management Centre, Andrew Campbell, Ashridge Strategic, and Management Centre. Work : Creating Clarity on Unit Roles and Responsibility. 21(3):351–363, 2003.

[31] G Scott Graham and Peter J Denning. Protection. In *Proceedings of the November 16-18, 1971, fall joint computer conference on - AFIPS '71 (Fall)*, page 417, New York, New York, USA, 1971. ACM Press.

[32] Charles B Haley, Jonathan D Moffett, Robin Laney, and Bashar Nuseibeh. A Framework for Security Requirements Engineering. *Proceedings of the 2006 international workshop on Software engineering for secure systems, SESS'06*, pages 35–41, 2006.

[33] Qingfeng He and AI Antón. A framework for modeling privacy requirements in role engineering. *Proc. of REFSQ*, 2003.

[34] Vincent C. Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, jan 2014.

[35] Edward Humphreys. Information security management standards: Compliance, governance and risk management. *Information Security Technical Report*, 13(4):247–255, 2008.

[36] John Hunt. *Agile software construction.* 2006.

[37] Julie Ireton. Phoenix payroll system doomed from the start, 2017.

[38] Pooya Jaferian and Konstantin Beznosov. Poster : Helping users review and make sense of access policies in organizations. In *SOUPS '14: Proceedings of the Tenth Symposium On Usable Privacy and Security*, pages 301–320, 2014.

[39] Xin Jin, Ravi Sandhu, and Ram Krishnan. RABAC: Role-centric attribute-based access control. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7531 LNCS:84–96, 2012.

[40] Alan H Karp, Harry Haury, and Michael H Davis. From ABAC to ZBAC : The Evolution of Access Control Models. *ISSA Journal*, (April):22–30, 2010.

[41] Lawrence Katz and Robert Margo. Technical Change and the Relative Demand for Skilled Labor: The United States in Historical Perspective. Technical Report January, National Bureau of Economic Research, Cambridge, MA, feb 2013.

[42] Axel Kern, Andreas Schaad, and Jonathan Moffett. An administration concept for the enterprise role-based access control model. *Proceedings of the eighth ACM symposium on Access control models and technologies - SACMAT '03*, page 3, 2003.

[43] Angelos D. Keromytis and Jonathan M. Smith. Requirements for scalable access control and security management architectures. *ACM Transactions on Internet Technology*, 7(2):8–es, 2007.

[44] M. Fahim Ferdous Khan and Ken Sakamura. Fine-grained access control to medical records in digital healthcare enterprises. *2015 International*

Symposium on Networks, Computers and Communications, ISNCC 2015, 2015.

[45] D. Richard Kuhn, Edward J. Coyne, and Timothy R. Weil. Adding attributes to role-based access control. *Computer*, 43(6):79–81, 2010.

[46] Wouter Kuijper and Victor Ermolaev. Sorting out role based access control. *Proceedings of the 19th ACM symposium on Access control models and technologies - SACMAT '14*, pages 63–74, 2014.

[47] Irwin Kwan, Marcelo Cataldo, and Daniela Damian. Conway's Law Revisited: The Evidence for a Task-Based Perspective. *IEEE Software*, 29(1):90–93, jan 2012.

[48] Romain Laborde, François Barrère, and Abdelmalek Benzekri. Toward authorization as a service: a study of the XACML standard. *Communications and Networking Symposium*, page 9, 2013.

[49] Butler W Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, jan 1974.

[50] Ulrich Lang. OpenPMF SCaaS: Authorization as a service for cloud & SOA applications. *Proceedings - 2nd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2010*, pages 634–643, 2010.

[51] Ninghui Li and Ziqing Mao. Administration in role-based access control. *Proceedings of the 2nd ACM symposium on Information, computer and communications security - ASIACCS '07*, page 127, 2007.

[52] Haibing Lu, Yuan Hong, Yanjiang Yang, Lian Duan, and Nazia Badar. Towards user-oriented RBAC model. *Journal of Computer Security*, 23(1):107–129, 2015.

[53] Thomas W Malone and Kevin Crowston. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.

[54] Michelle L Mazurek, J P Arsenault, Joanna Bresee, Nitin Gupta, Iulia Ion, Christina Johns, Daniel Lee, Yuan Liang, Jenny Olsen, Brandon Salmon, Richard Shay, Kami Vaniea, Lujo Bauer, Lorrie Faith Cranor, Gregory R Ganger, Michael K Reiter, and E T H Z. Access Control for Home Data Sharing : Attitudes , Needs and Practices. *Computing*, 48(2):645–654, 2010.

[55] Barsha Mitra, Shamik Sural, Jaideep Vaidya, and Vijayalakshmi Atluri. A Survey of Role Mining. *ACM Computing Surveys*, 48(4):1–37, 2016.

[56] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering. In *Proceedings of the conference on The future of Software engineering - ICSE '00*, pages 35–46, New York, New York, USA, 2000. ACM Press.

[57] Matunda Nyanchama and Sylvia Osborn. The role graph model and conflict of interest. *ACM Transactions on Information and System Security*, 2(1):3–33, feb 1999.

[58] AC O'Connor and RJ Loomis. 2010 Economic Analysis of Role-Based Access Control. Technical Report 0211876, NIST, 2010.

[59] Government of Canada. Operational Security Standard: Management of Information Technology Security (MITS), 2017.

[60] Government of Canada. Policy on Government Security, 2017.

[61] Government of Canada. Policy on Information Management, 2017.

[62] Sejong Oh and Seog Park. Task-role based access control (T-RBAC): An improved access control model for enterprise environment. *Database and Expert Systems Applications*, pages 264–273, 2000.

[63] Sejong Oh and Ravi Sandhu. A model for role administration using organization structure. *Proceedings of the seventh ACM symposium on Access control models and technologies - SACMAT '02*, page 155, 2002.

[64] Sejong Oh, Ravi Sandhu, and Xinwen Zhang. An effective role administration model using organization structure. *ACM Transactions on Information and System Security*, 9(2):113–137, may 2006.

[65] Päivi Ovaska, Matti Rossi, and Pentti Marttiin. Architecture as a coordination tool in multi-site software development. *Software Process Improvement and Practice*, 8(4):233–247, 2003.

[66] Sean Peisert and Matt Bishop. Dynamic, Flexible, and Optimistic Access Control. Technical report, UC Davis CS Technical Report CSE-2013-76, 2013.

[67] Raghuram G Rajan and Julie Wulf. the Flattening Firm: Evidence From Panel Data on the Changing Nature of Corporate Hierarchies. *Review of Economics & Statistics*, 88(4):759–773, 2006.

[68] Syed Zain R Rizvi and Philip W L Fong. Interoperability of Relationship- and Role-Based Access Control. *Codaspy*, pages 231–242, 2016.

[69] Syed Zain R. Rizvi, Philip W.L. Fong, Jason Crampton, and James Sellwood. Relationship-Based Access Control for an Open-Source Medical Records System. *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies - SACMAT '15*, pages 113–124, 2015.

[70] Karina Roman. Federal government to downsize failing Canada.ca project, 2017.

[71] R. Sandhu and Q. Munawer. The ARBAC99 model for administration of roles. In *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*, pages 229–238. IEEE Comput. Soc, 1999.

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

[73] Ravi Sandhu and Venkata Bhamidipati. Role-based administration of user-role assignment: The URA97 model and its Oracle implementation. *Journal of Computer Security*, 1999.

[74] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security*, 2(1):105–135, feb 1999.

[75] Ravi Sandhu and Qamar Munawer. How to do Discretionary Access Control Using Roles. *Proceedings of the third ACM Workshop on Role-Based Access Control (RBAC '98)*, pages 47–54, 1998.

[76] Andreas Schaad, Jonathan Moffett, and Jeremy Jacob. The role-based access control system of a European bank. In *Proceedings of the sixth ACM symposium on Access control models and technologies - SACMAT '01*, pages 3–9, New York, New York, USA, 2001. ACM Press.

[77] Monique Scotti. Government email project still stalled after five years $100 million spent, 2017.

[78] Daniel Servos and Sylvia L. Osborn. Current Research and Open Problems in Attribute-Based Access Control. *ACM Computing Surveys*, 49(4):1–45, 2017.

[79] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1–11, 2011.

[80] San-Tsai Sun and Konstantin Beznosov. The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems. *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*, pages 378–390, 2012.

[81] Bo Tang, Ravi Sandhu, and Qi Li. Multi-tenancy authorization models for collaborative cloud services. *Concurrency Computation*, 27(11):2851–2868, 2015.

[82] Treasury Board of Canada Secretariat. Government of Canada Strategic Plan for Information Management and Information Technology 2017 to 2021, 2017.

[83] Jaideep Vaidya, Vijayalakshmi Atluri, Janice Warner, and Qi Guo. Role engineering via prioritized subset enumeration. *IEEE Transactions on Dependable and Secure Computing*, 7(3):300–314, 2010.

[84] Shin Jer Yang, Pei Ci Lai, and Jyhjong Lin. Design role-based multi-tenancy access control scheme for cloud services. *Proceedings - 2013 International Symposium on Biometrics and Security Technologies, ISBAST 2013*, (1):273–279, 2013.

[85] J. Yong, E. Bertino, M. Toleman, and D. Roberts. Extended RBAC with role attributes. *PACIS 2006 - 10th Pacific Asia Conference on Information Systems: ICT and Innovation Economy*, 2006.

[86] Younis A. Younis, Kashif Kifayat, and Madjid Merabti. An access control model for cloud computing. *Journal of Information Security and Applications*, 19(1):45–60, 2014.

[87] Hadar Ziv and Debra J Richardson. The Uncertainty Principle in Software Engineering. *19th International Conference on Software Engineering*, pages 1–20, 1997.