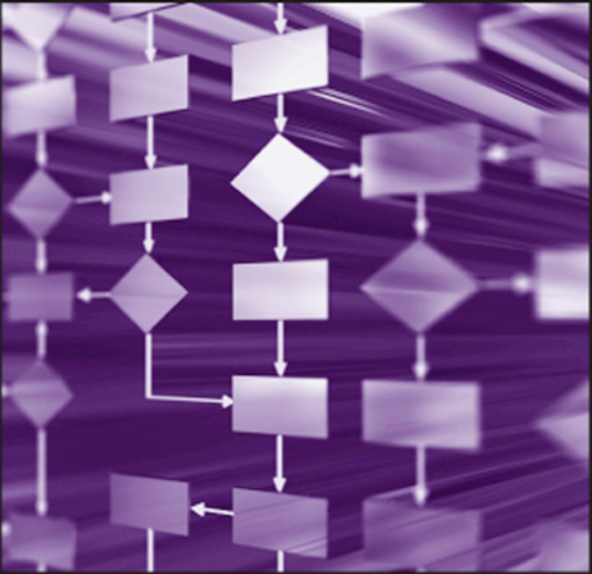
ARTECH HOUSE

INFORMATION SECURITY AND PRIVACY SERIES

Role Engineering *for Enterprise Security Management*



EDWARD J. COYNE • JOHN M. DAVIS

Role Engineering for Enterprise Security Management

Role Engineering for Enterprise Security Management

Edward J. Coyne
John M. Davis

ARTECH HOUSE
BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the U.S. Library of Congress.

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library.

Cover design by Yekaterina Ratner

ISBN 13: 978-1-59693-218-0

10 9 8 7 6 5 4 3 2 1

Contents

Preface

Role-based access control (RBAC) promises to provide several benefits to organizations. These benefits include simplified security provisioning and administration, ease of reporting on privileges and to whom they are available, and finer grained security authorization. By being policy-neutral, RBAC can be used to enforce the variety of access control policies that various organizations already have in place or may develop in preparation for the adoption of RBAC. RBAC also provides specific features to facilitate implementation of access control policies. These features include capabilities to impose constraints on relationships among roles and among the components of roles, and the inheritance of permissions from one role by another that can simplify role design.

To employ RBAC it is first necessary to identify a set of roles for the organization. These roles must accurately reflect the activities, functions, and responsibilities within the organization. Roles have two major components: the names of the job functions performed by IT users, and the permissions that enforce an access control policy. The definition of roles is a process of discovering and then engineering requirements for access control. A methodology for establishing a valid set of role names with assigned permissions is needed. This book is designed to assist organizations in establishing such a role engineering methodology before starting a role engineering effort. Previous practical experience is applied to provide practical guidance in defining roles and in structuring the roles for use in controlling access to IT resources. We provide guidance on using role engineering tools to assist in carrying out the methodology and approaches to staffing and team building that help organizations achieve quality results with efficient use of resources.

Role engineering is an approach to defining roles and assigning permissions to the roles. It must capture the organization's security rules and business rules, as these relate to access control. These rules must be reflected in defining, naming, structuring, and constraining a valid set of roles. Role engineering covers the design of all components of the RBAC models except for assignment of individual users to roles. These models include role names, permissions, and constraints, and the roles may be structured into hierarchies of permission inheritance.

We will describe what role engineering is and how it should be conducted to achieve success in both quality of results and efficient use of resources. By identifying key factors for successful role engineering and providing practical approaches based on experience, this book can help organizations avoid a less-than-optimal role engineering effort. We further point out benefits that role engineering itself can provide whether or not RBAC is adopted as an access control model for a given organization.

The book can assist with the process of justification of a role engineering project based on return on investment. It also provides guidance on project planning. Planning topics include determining scope, estimating costs and schedules, weighing risks of decisions in the role engineering process, staffing the effort, measuring progress, recording results, and identifying next steps.

As part of the process of developing staff to carry out role engineering projects, the book provides material that may be used to support the training of role engineering practitioners. A focus on goals and guidelines for achieving the goals can guide role engineering staff in the most important aspects of role engineering.

Insight into some current research on role engineering is provided to highlight some areas where additional development will be required to enhance the understanding of role engineering in the complex environments that are found in practice.

Finally, the book provides guidance on metrics and testing that can help ensure the success of RBAC implementation in the enterprise. By monitoring and reviewing progress and results using metrics, the effort can be kept on track and productive.

The principles provided here can be used to prepare and execute individual project plans. The book thus provides a firm foundation on which each enterprise can build its own role engineering capability.

Intended Audience

The primary audience for this book is envisioned to be the members of a team that will be responsible for organizing and conducting a role engineering effort in an enterprise. The full range of planning, justification, and scoping of the effort is addressed. An underlying process of continuous improvement in approaches and techniques is recommended for role engineering. This continuous improvement process will also require staff with skills in software engineering, program management, and economic analysis.

The book does not assume that the reader will be proficient in information technology at a high technical level. While some aspects of the role engineering process involve use of information systems and analysis techniques, much of the role engineering process described involves more of a subject matter understanding of an organization's jobs and associated job functions and how these relate to the use of information technology.

Acknowledgments

The authors express deep appreciation for the kind help received from the following individuals and companies in reporting practical information on role engineering processes, efforts, and results: Ron Ryman of Eurekify, Inc., Martin Kuhlmann and René Weil of Beta Systems Software of America, Inc., Camilla Odenteg and her colleagues at Generic Integration AB, Joe Towne of Vaau, Inc., and the U.S. Veterans Health Administration.

1

Introduction

Background for the Book

This book brings together the essential elements needed to establish role-based access control (RBAC) within an enterprise. As enterprises in fields such as healthcare, law, government, industry, commerce, manufacturing, and services establish appropriate policies and controls, these must be developed in accordance with the organizations' mission, objectives, vision, and activities. The current norms for the production of goods and services define the achievement of results and indicate activities that must be addressed when establishing access control within information technology (IT) systems.

In controlling the ability for system users to access the various resources available on a network or system, it must be possible to identify each user and to set up automated rules, called *permissions*, that control what a user is allowed to access and how that access may take place. Thus it is necessary to assign permissions to users. This is a simple matter in principle. However, when there are a large number of users and a large number of permissions available, the administrative burden of assigning permissions to users becomes excessive. With this, it is difficult to track and audit just what permissions a user has and what users have been assigned a given permission.

With RBAC, permissions are assigned to roles instead of to users. This creates a layer of abstraction where users can be assigned to roles instead of permissions being assigned directly to these users. Because the number of roles in an organization is usually much smaller than the number of permissions in that organization's IT systems and networks, the layer of abstraction provided by RBAC can simplify the authorization of permissions to the users. Also, roles and permissions typically change much more slowly over time than do personnel, which is one reason why RBAC is effective in reducing administration costs. Users are assigned to the roles and they automatically received all the permissions associated with the assigned roles. It becomes a relatively simple matter for a person to be assigned the roles they need without the necessity of the assigner even understanding what the permissions are on the various networks or systems. That work will have been done once by software engineers. Once roles with their permissions have been defined, an administrative rather than a technical person can perform the assignment of users to roles. This is a major benefit of RBAC. The advantages that RBAC can provide include reduced administrative costs, support for finer-grained access control policies, and support for auditing and reporting on authorizations of users to access corporate resources.

Once this principle of RBAC has been adopted, additional benefits are also made available. The role abstraction can be manipulated and designs can be created to implement constraints among roles and other RBAC components as well as hierarchies of roles with inheritance of permissions. These constraints and hierarchies can provide a high degree of granularity in supporting the organization's access control policy.

Role engineering is the definition of roles and their structures. It is essentially a process of gathering and analyzing requirements. Implementation of RBAC entails establishment of an IT infrastructure to represent and evaluate roles and to provide access control decisions to consuming applications and other IT resources. This RBAC infrastructure, in turn, entails the establishment of role definitions. Thus, role engineering is a necessary adjunct to RBAC. What exactly is RBAC? The book by Ferraiolo, Kuhn, and Chandramouli [1] provides a comprehensive treatment of the entire subject of RBAC. Other works provide the basics of RBAC [2, 3]. Definitions for the concepts and constructs of RBAC are provided by the RBAC Standard [4, 5].

This book provides guidance and recommendations on conducting a role engineering effort. While understanding the subject of RBAC itself can require some technical background, understanding role engineering for the most part does not require a technical background. That part of role engineering dealing with assigning actual IT permissions to roles does require technical expertise, but understanding and managing the process of assigning these IT permissions to roles does not. The book will assist you in defining the roles that will enable your organization to use RBAC effectively. A combination of methodology with rationale, practical guidance, awareness of efforts to date and their results, and worked examples is provided to get you started along the path to developing a set of roles for your organization.

The book describes how software tools can assist in the process of defining an organization's roles. One class of these tools captures an organization's current permissions from their systems and records the users to whom the permissions have been assigned. Using this data, these tools assist in clustering similar sets of access permissions into candidate roles. Other technologies treated in the book deal with the analysis of user and permission data from scenarios that represent real-world use of a system or systems. These technologies are in the realm of data modeling and analysis—disciplines that are mature and in daily use in other settings.

Role-Based Access Control

To promote the development of the ideas and technologies needed to advance the state of the art of RBAC, in the mid-1990s the National Institute of Standards and Technology (NIST) selected SETA Corporation of McLean, Virginia, as an industry partner. This partnership, together with the Association for Computing Machinery (ACM), resulted in the inauguration of the annual RBAC Workshops (ACM Workshop on Role-Based Access Control; later the name was changed to ACM Symposium on Access Control Models and Technologies—SACMAT). These workshops, in turn, provided a rich source of research papers in RBAC technologies and related matters, such as role engineering. Dr. Ravi Sandhu, then a professor at George Mason University and an employee of SETA, led the preparation of a series of papers defining RBAC and investigating many facets of RBAC. One of these papers formed the nucleus of the present RBAC Standard [2, 4, 6]. According to this paper,

> A role is chiefly a semantic construct forming the basis of access control policy. With RBAC, system administrators create roles according to the job functions performed in a company or organization, grant permissions (access authorization) to those roles, and then assign users to the roles on the basis of their specific job responsibilities and qualifications….

NIST continues to provide leadership in the development, adoption, and standardization of RBAC. Their Web site serves as a valuable repository for RBAC and role engineering information [7].

Role Engineering

The term "role engineering" dates back to the mid-1990s [8]. The term is analogous to the business process reengineering discipline that was current at that time. Business process reengineering, as does role engineering, involves analysis, synthesis, testing, building blocks, and construction, and it results in a usable structure.

The definition of "business process reengineering" sponsored by SearchCIO.com, powered by WhatIs.com, an online computer dictionary [9], is as follows:

> Business process reengineering (BPR) is the analysis and redesign of workflow within and between enterprises. BPR reached its heyday in the early 1990's when Michael Hammer and James Champy published their best-selling book, "Reengineering the Corporation". The authors promoted the idea that sometimes radical redesign and reorganization of an enterprise (wiping the slate clean) was necessary to lower costs and increase quality of service and that information technology was the key enabler for that radical change. Hammer and Champy felt that the design of workflow in most large corporations was based on assumptions about technology, people, and organizational goals that were no longer valid. They suggested seven principles of reengineering to streamline….

This is the origin of the term "engineering" in role engineering. Beyond this origin, the process of developing a structure of roles for an enterprise does involve engineering in the traditional sense. Further, the term also defines the process as a pragmatic discipline that is differentiated from a science. In engineering we typically find a process of defining a baseline for a design and subsequently a process of refinement and optimization to improve on the baseline. Thus, it is the result that is of importance and not so much the means used to arrive at the result.

The Canadian Academy of Engineering definition of "engineering" is as follows [10]:

> Engineering is a profession concerned with the creation of new and improved systems, processes and products. The central focus of engineering is design, an art entailing the exercise of ingenuity, imagination, knowledge, skill, discipline and judgement based on experience.

So role engineering is the application of engineering principals and techniques to create a set of roles that implements a security policy and that is organized into a structure that reflects the nature of the enterprise or organization. The role structure will be optimized for effectiveness and efficiency using engineering principles and techniques.

Aims of the Book

The aims of this book are to define what role engineering is, elucidate the role played by role engineering, identify key factors in successful role engineering, provide practical approaches to role engineering, and provide information on completed and ongoing role engineering efforts. To accomplish these aims, the book provides a summary of RBAC, establishes of a set of goals of role engineering, and describes various methods and tools for role engineering.

It may be assumed that role engineering would only be performed if there is the intention of adopting RBAC as a mechanism for implementing access control in an enterprise. There may be cases, however, where role engineering would be advantageous even if RBAC is not to be adopted as such a mechanism. In these cases, RBAC can serve as a model for access control while actual implementation of the mechanism for access control is not RBAC, at least not in totality. Using RBAC as an access control model and conducting a role engineering effort will result in the identification of the organization's access control policy and a set of requirements for implementing that policy.

How the Book Can Be Used

This book can assist with the process of justification of an RBAC and/or a role engineering project. It is suitable for use as guidance for project planning. Once it has been decided to proceed with a role engineering effort, guidance in this book can assist in the planning process. Planning topics include determining scope, estimating costs and schedules, weighing risks of decisions in the role engineering process, staffing the effort, measuring progress, recording results, and identifying next steps.

The book can be used as a source for preparing process descriptions. Each role engineering effort will typically have its own tailored process to be followed. By providing components of processes and guidelines for combining these processes, the reader will be able to assemble a suitable role engineering process for the planned role engineering effort.

As part of the process of developing staff to carry out role engineering projects, the book provides material that may be used to support the training of role engineering practitioners. A focus on goals and guidelines for achieving the goals can guide role engineering staff in the most important aspects of role engineering.

The book is designed to help the reader to sort out the various aspects of role engineering, and RBAC in general. Using a statement of access control policy to identify permissions and constraints to name roles, to attach appropriate permissions to roles, to place constraints on uses, roles, and permissions, and to create role hierarchies, the role engineering practitioner can develop an effective and efficient role structure for the enterprise.

This book provides guidance on metrics and testing that can help ensure the success of RBAC implementation in the enterprise. By monitoring and reviewing progress and results using metrics, the effort can be kept on track and productive. Testing of interim results against the goals of the effort can ensure the quality of the results and permit any needed corrections to be made during the role engineering process, when they are easier and more cost effective to make than in later stages of the SDLC.

This book provides tested processes for deriving role definitions and illustrates potential problems and corresponding solutions. These processes are effective in developing a suitable role structure, but this is only part of the need to be fulfilled. As with most projects, control is needed to keep the application of valuable resources focused on the most important aspects of the project in order to ensure that the goals are adequately met within the allocated resources. Experience on role engineering efforts illustrates that there are some typical pitfalls to be avoided. True success for a role engineering effort depends primarily on the results obtained, but it is also necessary to control the resources consumed to obtain the results.

The book deals primarily with principles that can be used to prepare and execute individual project plans, rather than attempting to be a bible that is to be followed prescriptively. It is intended to provide a foundation upon which each enterprise can build its own role engineering capability.

In Chapter 2 we provide some considerations regarding the development and analysis of business cases regarding whether to engage in a role engineering effort, or, if a role engineering effort has been justified, defining the scope and expected results from the effort.

In Chapter 3 we place role engineering into a typical system development life cycle (SDLC). Dependencies between the availability of role engineering results and other system development activities are explored.

Chapter 4 introduces role engineering in the context of RBAC, access control policy, and system development. It includes a discussion of how access control policy is used to define roles.

Chapter 5 begins to enter into the specific process of role engineering by describing what is considered to be a good role (i.e., a role that serves the purpose of supporting the organization's access control policy in an efficient and effective manner).

Chapter 6 describes and evaluates approaches to defining roles. General principles are provided that can be applied to the particular circumstances of an organization's needs and expectations for RBAC. Chapter 7 delves more deeply into the actual designing of roles, including role names, permissions, constraints, and hierarchies. Chapter 8 provides detail on the engineering of permissions to be assigned to role names.

In Chapter 9 we identify some tools that may prove useful in the role engineering process. Some of these products serve as repositories of engineering role components, while others incorporate specific methods such as those for role mining in a bottom-up discovery approach.

Chapter 10 provides guidance on putting the various pieces together to create roles with, possibly, constraints and hierarchies.

Chapter 11 provides case studies on what various organizations have encountered and accomplished in their role engineering efforts. Lessons learned from these experiences may be useful in illustrating principles for conducting a successful role engineering effort and possibly for illustrating potential pitfalls that are to be avoided.

Chapter 12 provides material to assist in planning a role engineering effort, and as part of this, Chapter 13 provides information on recommended staffing for the effort. Related to planning for role engineering, Chapter 14 points out some potential pitfalls to be avoided in the planning as well as execution phases of role engineering.

Finally, Chapter 15 summarizes and concludes the book.

The Bibliography at the end of the book includes a list of research papers, books, and standards references.

References

[1] Ferraiolo, D. F., D. R. Kuhn, and R. Chandramouli, *Role-Based Access Control*, 2nd ed., Norwood, MA: Artech House, 2007.

[2] Sandhu, R., et al., "Role-Based Access Control Models," *IEEE Computer*, Vol. 29, No. 2, February 1996, pp. 38–47.

[3] Ferraiolo, D. F., and D. R. Kuhn, "Role Based Access Control," *15th National Computer Security Conference*, Baltimore, MD, 1992, pp. 554–563.

[4] InterNational Committee for Information Technology Standards (INCITS), ANSI INCITS 359-2004 American National Standard for Information Technology—Role Based Access Control, February 3, 2004.

[5] Ferraiolo, D. F., et al., "Proposed NIST Standard for Role-Based Access Control," *ACM Transactions on Information and System Security*, Vol. 4, No. 3, August 2001, pp. 224–274.

[6] Sandhu, R., D. F. Ferraiolo, and D. R. Kuhn, "The NIST Model for Role Based Access Control: Towards a Unified Standard," *Proceedings of the 5th ACM Workshop on Role Based Access Control*, Berlin, Germany, July 26–27, 2000, pp. 47–63.

[7] NIST-RBAC, http://csrc.nist.gov/rbac/.

[8] Coyne, E. J., "Role Engineering," *Proceedings of the 1st ACM Workshop on Role-Based Access Control*, Gaithersburg, MD, November 30–December 1, 1995, pp. I-15–I-16.

[9] Business Process Reengineering, http://www.bitpipe.com/rlist/term/Business-Process-Reengineering.html.

[10] IEEE EWH, http://www.ewh.ieee.org/cmte/pa/Status/Engineering.html.

2

The Business Case for Role-Based Access Control

In many instances, especially in the area of IT security, the availability of technology drives how security is implemented. Product vendors seem to be continually touting the capabilities of their products and stating why these should be acquired. In many cases there is a belief on the part of the consumer that security requirements can be met by simply acquiring and installing a product. Of course, this is not a meaningful way to ensure security requirements because the needs of each enterprise are different and require specific tailoring of security products based on local analysis. Prior to determining the needs of the enterprise and tailoring security products to these needs, a business case should be made to justify the time and resources that will be needed to carry out these activities.

RBAC has been gaining in popularity among IT practitioners and managers. Its promise of simplifying the administration of security authorizations and its ability to enforce any number of security policies make it a technology that deserves serious consideration for adoption for an enterprise. In fact it has been selected for adoption in many enterprises typically along with other approaches for implementing access control.

Evaluating the RBAC Business Case

Whether or not RBAC is adopted should depend on the results of a comprehensive survey and evaluation of available technologies in the security administration and enforcement problem space. Part of this evaluation should be the approaches and corresponding resources required for role engineering. This goes beyond the identification and selection of technology; role engineering costs can exceed the acquisition and maintenance costs of RBAC technology itself.

Data gathering and analysis are necessary to gain an understanding of the security needs of the enterprise and to identify what technologies and products should be considered for acquisition and deployment to meet the security requirements. Guiding the data analysis and evaluation process should be a set of security requirements. Without these the evaluation and acquisition process will lack a firm foundation. So creation, adoption, and identification of a set of security requirements are needed in making the business case for RBAC. In the absence of an understanding of what the security deficiencies are (however these deficiencies may be defined), the selection of technologies and products will necessarily be haphazard. Furthermore, when these products are implemented, their appropriate configuration will probably not be an optimal one. Therefore, as one of these security technologies, RBAC should be used only where there is a business case to justify it. Figure 2.1 illustrates a requirements-driven decision to implement security technologies and products, RBAC products among them.

Security Requirements

Security requirements are also needed to establish the security policy that roles will support. Thus they are a prerequisite of role engineering. These security requirements can be the result of risk analysis, mandates from government or other authorities, or may be industry best practices whose implementation may be considered as performing due diligence with regard to IT security.

Given that the enterprise is committed to the use of RBAC, the business case for a role engineering effort will, to a large extent, derive from a business case made for the use of RBAC. In many cases, the decision to use RBAC will have been made by the time a role engineering effort is ready to begin. However, part of establishing a business case for RBAC may itself entail a thorough understanding of role engineering and its resource

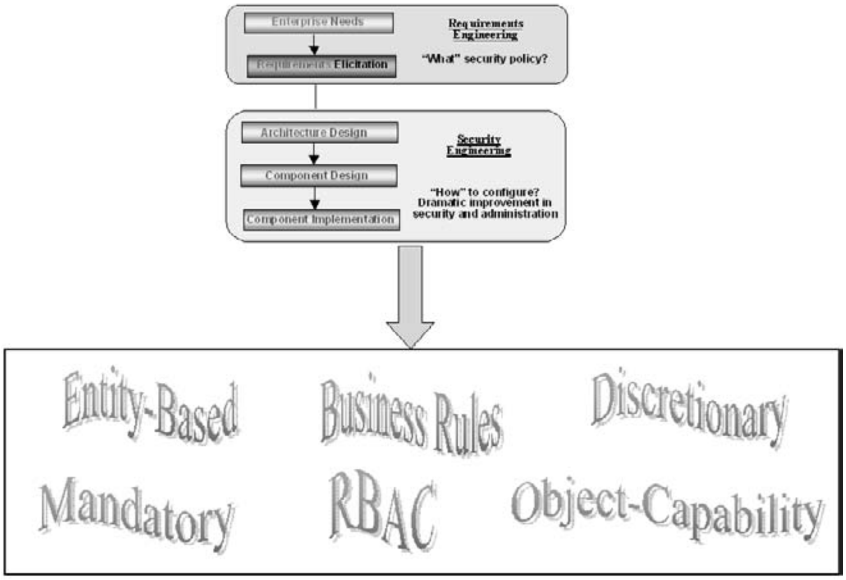


Figure 2.1 Requirements-driven decision for security implementation.

requirements. Here we discuss the economic case, the security case, and the compliance case for the role engineering effort and, to some extent, the overall RBAC implementation.

Return on Investment

To estimate the return on investment of a role engineering effort, it is necessary to estimate the costs of conducting the effort and to weigh these against the anticipated benefit. Some rules of thumb for making the cost estimate are provided in Chapter 13. Additional information is available in the report *The Economic Impact of Role-Based Access Control* [1], which is available on the NIST RBAC Web site (http://www.nist.gov/director/prog-ofc/report02.1.pdf). For example, in this report, benefits of RBAC include some benefits of role engineering itself:

- Simplified systems administration;
- Enhanced organizational productivity;

- Reduction in new employee downtime;
- Enhanced systems security and integrity;
- Simplified regulatory compliance.

To this list we would add enhanced security. When roles have been well engineered (i.e., permissions have been assigned to roles to provide exactly the access required by a holder of the role), the security principle of least privilege will be met. This, accompanied by RBAC's ease of assigning and revoking roles to and from users will result in a capability to enforce an access control policy precisely and continually over time.

Well-engineered roles in an RBAC environment can serve to reduce risk in the enterprise because of this possibility of precision in assigning permissions to users through roles. The major risk addressed is the common one where the permissions assigned directly to a given user are only poorly documented and not understood in their entirety. Compounding this lack of awareness of a user's total permissions on all systems is the fact that over time a user will be typically assigned an ever-growing set of permissions, simple because it is so daunting a task to keep up with the user's actual needs as job assignments change. With RBAC, where permissions are carefully assigned to management-approved roles, it is always well known what set of permissions belongs to a given role. So long as users are assigned only to those roles they need, and are revoked from those roles when the need is no longer present, control over user-permission assignments will be much higher that would be the case without RBAC. This, in turn will reduce the risk of excessive permission assignment to a user with its resultant security vulnerability.

We note that role engineering is involved in *enhanced organizational productivity*. In [1], regarding enhanced organizational productivity, it is noted that:

> RBAC also has the potential to enhance the system by which firms and organizations structure their information systems. Because of the greater flexibility and breadth of network design associated with RBAC, the model can be adapted to mirror the organizational structure. This creates the potential for new and innovative ways of structuring the organization, altering the routing of information, or changing the organization's production processes. Organizations can benefit from the consistency in infrastructure across divisions or units within the same entity. Additionally, improved business standards may result in cost savings. The synergistic improvements that may occur within a company could have potentially large impacts on employee productivity.

Thus, if role definitions effectively model organization structure, and also authority, workflow, and responsibility, the roles can potentially be used throughout an enterprise with the benefits of standard roles (the alternative would be to have differing access controls in different parts of the enterprise). Furthermore, these roles not only apply throughout the enterprise, but the same roles can be used across all system resources, including devices and applications. Figure 2.2 illustrates the aspects of roles that make them useful in reflecting these features of an enterprise.

Another benefit of well-engineered roles to the enterprise is an improved understanding of the structures and information flows which can lead to improvements in business processes and their interactions. Simply put, a by-product of the role engineering effort is a better understanding of how the enterprise and its various components function. This understanding can be leveraged to make improvements in business processes, which in turn can result in greater productivity and effectiveness. This benefit is illustrated in Figure 2.3.

What are the criteria for deriving maximum benefit from a role engineering effort? These may include:

- Defining and naming all of the enterprise's objects that warrant protection by the access control policy (realized as objects in permission definitions) to ensure completeness in the access control policy;

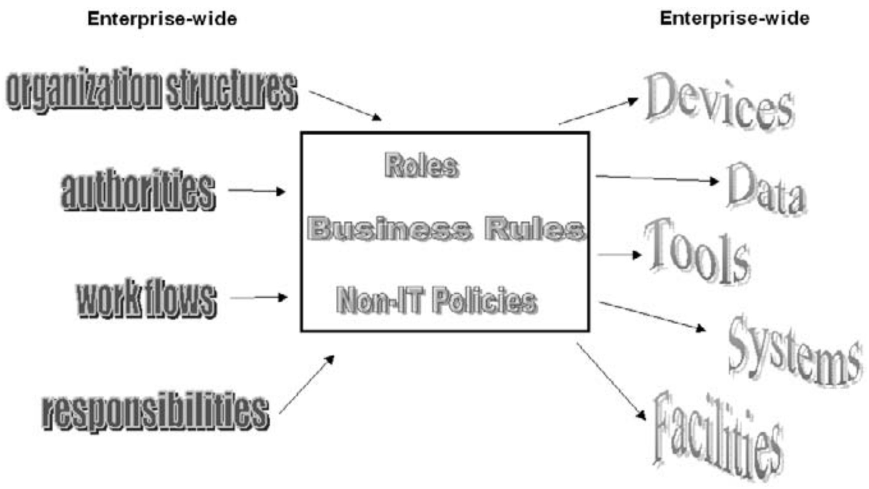
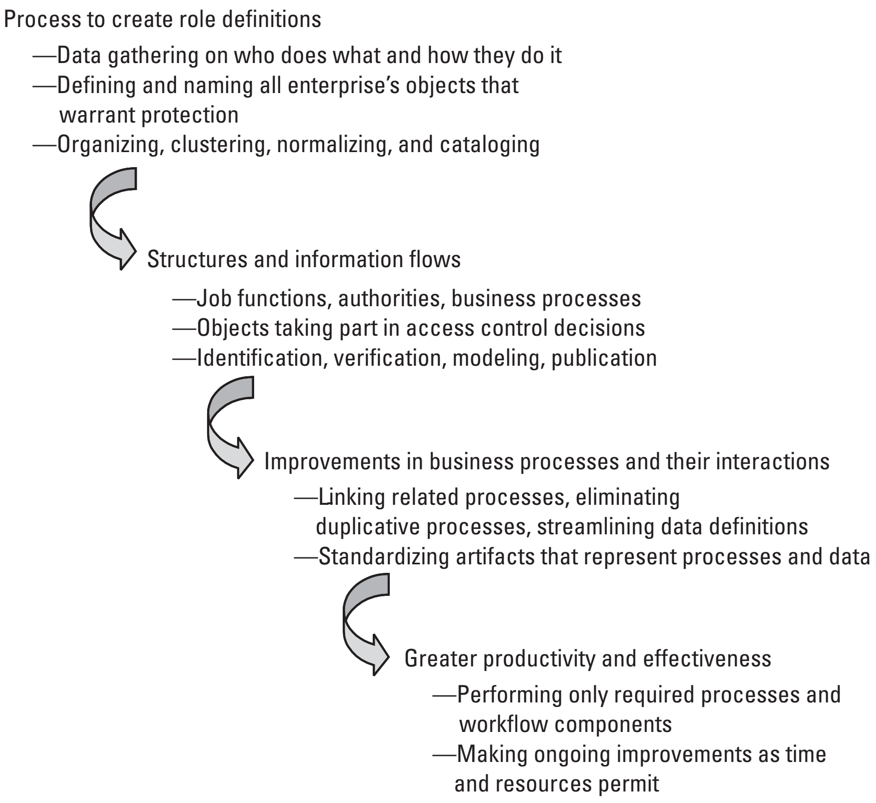


Figure 2.2 Enterprise aspects of roles.



Process to create role definitions
—Data gathering on who does what and how they do it
—Defining and naming all enterprise's objects that warrant protection
—Organizing, clustering, normalizing, and cataloging

Structures and information flows
—Job functions, authorities, business processes
—Objects taking part in access control decisions
—Identification, verification, modeling, publication

Improvements in business processes and their interactions
—Linking related processes, eliminating duplicative processes, streamlining data definitions
—Standardizing artifacts that represent processes and data

Greater productivity and effectiveness
—Performing only required processes and workflow components
—Making ongoing improvements as time and resources permit

Figure 2.3 Benefits of well-engineered roles.

- Defining all protected information flows and the channeling of the flows among individuals in the enterprise (realized by operations on objects, i.e., permissions) to ensure that rules to be included in the access control policy are complete;
- Defining and naming all job functions, authorities, and responsibilities (realized by actors, ultimately role names) as components of the access control policy.

If these criteria for complete definition of security-relevant objects, information flows, and actors are met, roles can be defined, and as a by-product, the enterprise's business functions can be documented and optimized. This will provide information to be used in efforts to improve effectiveness and efficiency and provide a baseline for all future improvements. The criteria are as follows:

- Defining and naming all the enterprise's objects that warrant protection by the access control policy;
- Defining all protected information flows and the channeling of the flows among individuals in the enterprise;
- Defining and naming all the job functions, authorities, and responsibilities.

Rules of thumb for estimating benefits are presented in Table 2.1. This guide to estimating benefits may be of value when developing a proposal to management for possible adoption of RBAC or when explaining to employees why the organization is transitioning to RBAC.

In the process of identifying a set of permissions to be assigned to role names, it may turn out that the existing permission definitions are

Table 2.1
Rules of Thumb for Estimating Benefits

Rule of Thumb	**Comment**
1. Reduce all estimates to dollar estimates.	If not possible to reduce benefits to dollar estimates, identify a list of benefits expressed in text (e.g., support enterprise architecture; baseline security requirements to be used for other things, such as development).
2. Consider the direct benefit(s) of employing RBAC supported by role engineering.	Benefits for RBAC trickle down to role engineering.
3. Consider side benefits, such as possible improvements in business processes enabled be role engineering efforts.	Side benefits could potentially be greater than the direct benefits. Any disadvantages to doing role engineering should be considered as costs, tangible or intangible.
4. Consider the case without the benefit.	One case without the benefit of role engineering might be a cost (tangible or intangible) attributable to inadequate or incorrect role definitions.
5. Consider related activities that will be executed anyway, providing synergy.	Perhaps an effort will be taking place anyway to baseline security requirements, plan for application replacement, organize an enterprise planning team, or something similar that could feed into the role engineering effort.

overlapping, inconsistent, incomplete, or just incorrect. Before roles are defined it will be necessary to clean up the problems occurring among the enterprise's permissions. This requirement will definitely benefit the enterprise, both in clarifying the de facto access control policy and in providing a basis for valid role definitions.

Figure 2.4 illustrates the process of resolving problems in permission definitions. In the figure, "overlapping" refers to sets of roles that have overlapping sets of permissions. This condition can be corrected by separating roles with overlapping permissions into separate roles without the overlap, reassigning permissions to the roles so that they do not contain overlapping sets of permissions, or by renaming roles or the permissions to remove the overlapping nature of the role definitions. The resulting roles will now be independent of each other. "Incongruous" refers to sets of roles that contain inconsistent sets of permissions or similar set of permissions with inconsistent role names. These conditions can be corrected by resolving the inconsistencies of the sets of permissions, by redefining the roles, or by keeping the same permission assignments and renaming the roles. The resulting roles will now be congruous as to their role names and permissions assignments. "Incomplete" refers to role names that do not represent the total access control policy or permissions that do not include all of the permissions included in the access control policy. These conditions can be corrected by rescoping (i.e., adding to the set of role names or to the permissions assigned to the existing roles names—or both). The resulting roles will be fully scoped with

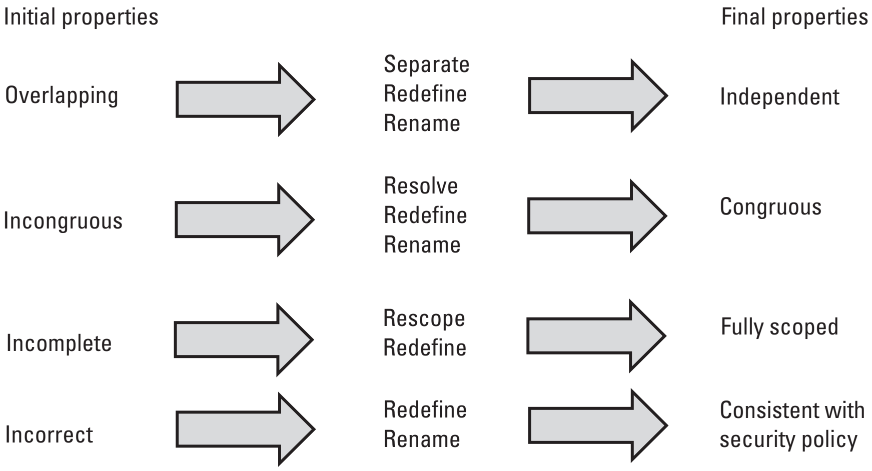


Figure 2.4 Resolving problems in permission definitions.

respect to the access control policy. “Incorrect” refers to role names or permissions assigned to the role names that conflict with the access control policy. These conditions can be corrected by redefining the permissions assigned to the existing roles or by renaming the roles to agree with the access control policy. The resulting roles will be consistent with the access control policy.

The Economic Case

In making an economic case it is necessary to identify areas for life-cycle cost savings and avoidance. For example, for a well-managed role engineering effort where adequate resources of the type needed are made available, it is possible to minimize errors and resulting rework in contrast to a low-key effort where results obtained are not commensurate with the effort and time expended. The economic case should be based on defining approaches for deriving maximum value from the anticipated role engineering efforts.

The potential benefits of the role engineering effort (as contrasted with the benefits of adoption of RBAC itself) are those that improve the business processes and functions (defining objects, information flows, and job functions). Therefore, the economic case will be based on these potential improvements and their effect on costs. Sources of cost savings resulting from this type of improvement include accelerated processing times that hasten cost recovery and profit accrual, more efficient assignment of personnel as workflows are streamlined, and facilitated assignment of personnel to job functions. The enterprise scope of the role definitions provides uniformity in definition of job functions, at least in relation to permissions in IT systems, and the ability to include all enterprise protected objects within the security policy wherever these are located in the organization.

The Security Case

In making the security case, the following considerations should be addressed: results of risk assessments, adherence to best practices, cost benefit trade-offs, assessment of existing security controls, and management capabilities.

Among the security best practices that may factor into the security case are the attention to least privilege in role definitions. This can be manifested both in role naming and in permission assignment. For example, different roles should be defined for different sets of permissions, where each role contains only the minimal set of permissions required by the corresponding role name. That is, the role names should be differentiated by specific job function or workflow and the permissions assigned to those roles should be different from each other and restricted in their assignments to roles.

When defining roles and permissions, the access control policy must be clearly defined and followed. In particular, it will be necessary to direct sufficient attention to potential interactions among roles, permissions, and users. For example, if a user can be assigned to two roles, each one of which does not violate the access control policy for the user, it may be possible that the combination of the two roles provides a set of permissions for the user that does violate the policy. Thus, in such a case there would need to be a constraint placed on the two roles that would be either static (administrative—do not assign these two roles to the same user) or dynamic (run time—do not permit a user to activate these two roles within a session). A similar constraint would exist where a policy of separation of duty is in effect, whereby the same user could not be assigned to two given roles or could not use two given roles in the same session.

The Compliance Case

In making the compliance case, the access control policy must be evaluated against the relevant compliance criteria. For example, for U.S. clinical applications the access control policy must comply with Health Insurance Portability and Accountability Act (HIPAA) criteria, U.S. federal government systems must comply with federal financial accounting standards, and in general enterprise systems of various types must comply with enterprise practices and requirements for both internal and business-to-business information processing. This implies an activity that determines the relevance of available compliance criteria and summarizes these for use in reviewing access control policy for adherence to the criteria. In turn, when the role definitions and their potential user assignments are tested and evaluated, if the results obey the access control policy, they will automatically meet the compliance criteria.

Related to the compliance case are various RBAC-related standards. These are described in Chapter 11. To provide some foundational information on RBAC standards that are referred to in other chapters, a summary is provided here. Table 2.2 identifies the major RBAC standards and provides some background information on them.

Table 2.2
Major RBAC Standards

Standard	**Description**	**Status**	**Relevance to Role Engineering**
ANSI INCITS 359-2004 American National Standard for Information Technology—Role Based Access Control	U.S. RBAC Standard— contains an RBAC reference model and an RBAC system and administrative functional specification	In force	RBAC reference model provides concepts and terminology for role definitions
ANSI INCITS Role Based Access Control Implementation Standard	U.S. RBAC Implementation Standard—interprets ANSI INCITS 359-2004 for compliance and interoperability	Draft—not balloted	RBAC reference model provides concepts and terminology for role definitions
OASIS XACML Profile for Role Based Access Control (RBAC)	Industry standard—provides XML language constructs to implement RBAC features of core, hierarchical, and dynamic separation of duty	In force	Once roles have been defined, XACML may be used to represent the resulting policy constructs
Health Level 7 (HL7) RBAC Role Engineering Process	Based on Neumann and Strembeck, *A Scenario-Driven Role Engineering Process for Functional RBAC Roles* [2]	Draft standard for trial use	Provides guidance on defining core RBAC roles
HL7 RBAC Healthcare Permission Catalog	Standard permissions for healthcare	Draft standard for trial use	Provides a standards set of healthcare permissions
HL7 Healthcare Scenario Roadmap	Standard permissions mapped to ASTM candidate roles	Draft standard for trial use	Illustrates a working set of permissions definitions and related candidate roles
HL7 RBAC Healthcare Scenarios	Healthcare scenarios for use in defining healthcare permissions	Draft standard for trial use	Illustrate scenarios that may be used to identify define roles
ASTM Standard Guide for Information Access Privileges to Health Information	List of healthcare-related role names	In force	Provides healthcare candidate role names for possible use in defining healthcare permissions

References

[1] Gallaher, M. P., A. C. O'Connor, and B. Kropp, *The Economic Impact of Role-Based Access Control*, Planning Report 02.1, National Institute of Standards and Technology, 2002.

[2] Neumann, G., and M. Strembeck, "A Scenario-Driven Role Engineering Process for Functional RBAC Roles," *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, Monterey, CA, June 3–4, 2002, pp. 33–42.

3

Role Engineering in the Phases of the System Development Life Cycle

Role-based access control as an enterprise-wide approach to security management is not necessarily limited to any set of applications or protected resources. Thus it is not expected that the enterprise role engineering effort would be limited to a pass through the system development life cycle (SDLC) of an application. However, as applications are developed there could be several role engineering efforts undertaken to provide the role definitions needed by the specific applications. This is certainly true with regard to planning activities. In practical terms, role engineering efforts could be included in the development plans of applications. If this is done, it will be advantageous to see where role engineering activities fit within the SDLC [1–3] to ensure that adequate attention is provided to role engineering planning and execution. Ultimately, the results of role engineering will benefit the whole enterprise, provided that each project-oriented effort is conducted with an eye toward achieving an enterprise-level result.

To achieve an enterprise-level result when role engineering efforts are conducted in conjunction with system development projects, it will be necessary to maintain a master repository of permission and role definitions. This repository must be designed to hold and maintain the role definitions in a consistent manner as the results of individual projects are added. The results of the individual project efforts will require reconciliation with the definitions in the enterprise-level repository, either as new entries are included or as a periodic activity. Since the reconciliation process results can be expected to influence the definitions made within each project, it is advisable to perform the enterprise-level reconciliation while the projects providing input are still underway. This will permit the timely change of project role definitions that result from the enterprise reconciliation process.

Whether conducted as an enterprise-wide activity or as a component of the system development life cycle, role engineering activities must be justified, planned, resourced, executed, and evaluated on a project basis. The questions to be addressed for role engineering are the questions of what?, who?, why?, when?, how?, and where?. Except for When?, these are addressed in Chapter 12. In this chapter we focus on the question of When?

Some influences on access control policy and, in turn, permissions in role definitions, are external to the enterprise. These external influences must be reflected in role definitions whether role engineering is conducted as an independent activity or whether it is conducted in conjunction with a system development effort. External influences may include legislation, interoperability requirements, and best practices established from common experience [4]. The direct effect of these influences on permissions involves the identification of protected objects and the operations that particular roles may execute on those objects. One of the possible indirect effects is an extension of the timeline of the effort to investigate and accommodate the external influences.

Conducting a Role Engineering Effort as an Independent Activity

Planning for role engineering should begin when a decision has been reached to implement a role-based access control approach to enterprise security. Closely related to the decision regarding possible use of RBAC will be a decision regarding the possible adoption of a service-oriented architecture, an identity management system, public key infrastructure, or similar enterprise-oriented assets that support security. The business decision to implement RBAC should include the weighing of an estimate of the required commitment of resources and time for role engineering.

Without adequate efforts to identify role engineering as an activity that will be needed to define roles and thereby capitalize on the benefits of RBAC, it may be easy to overlook or minimize the need for this work. The role engineering effort must be defined and sized and then proposed to management for a decision to go forward. Without management support and encouragement, it will be extremely difficult to conduct a successful role engineering effort.

Conducting a Role Engineering Effort in Conjunction with a System Development Effort

When conducted in conjunction with a system development project, role engineering will be carried out within the activities of an SDLC. Role engineering fits into the system development life cycle in several of its phases. For our purposes we consider the following SDLC phases:

- Initiation;
- Acquisition/development;
- Implementation;
- Operations and maintenance;
- Disposition.

Figure 3.1 illustrates role engineering–related activities to be performed at the enterprise level and during a system development life cycle.

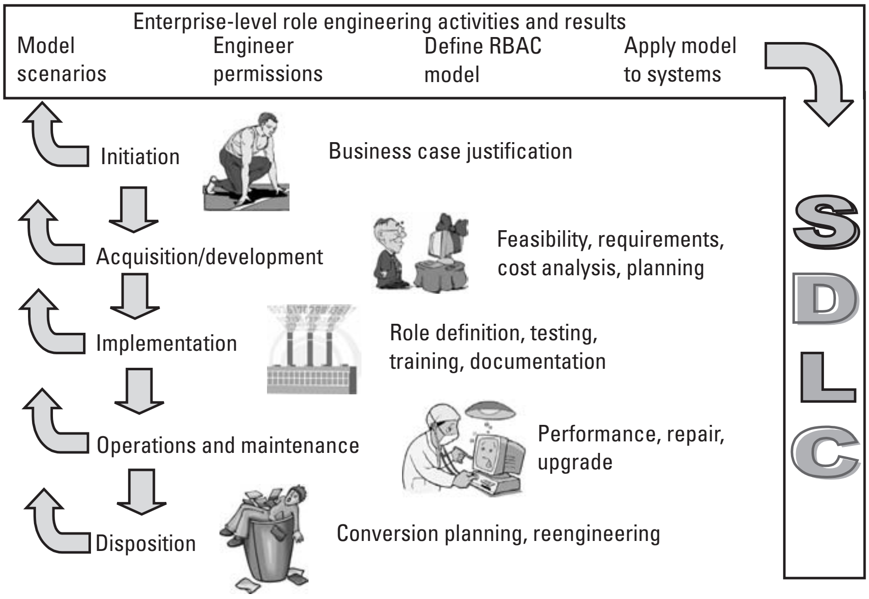

Enterprise-level role engineering activities and results
Model scenarios
Engineer permissions
Define RBAC model
Apply model to systems
Initiation
Business case justification
Acquisition/development
Feasibility, requirements, cost analysis, planning
Implementation
Role definition, testing, training, documentation
Operations and maintenance
Performance, repair, upgrade
Disposition
Conversion planning, reengineering
SDLC

Figure 3.1 Role engineering activities performed at the enterprise level and in the system development life cycle. (Clip art copyright clipart.com.)

Initiation Phase

In the initiation phase the need for RBAC itself will be identified and possibly approved. Possible sources of information to assist in determining a need for RBAC are provided in Chapter 2. Justification for the use of RBAC must recognize expenditures in staff and funds to establish an RBAC environment and must also include resources required for role engineering. These resources for role engineering include staff time to elicit role definitions from subject matter experts and from existing systems and tools, possibly role engineering tools, and possibly consulting services. Consulting services are closely related to role engineering tools, in that both may be acquired as a package. As a first order of magnitude, the major proportion of the role engineering costs may be considered to be the staffing costs. If no specialized tools or outside consultants are used, the in-house staffing will typically comprise about 90% of the total costs. If we suppose that specialized tools and outside consultants are used, in-house staffing could be reduced to 60% of the total and the tools and consulting services would comprise another 30% of the total. It is not advisable to acquire specialized role engineering tools for use by in-house staff with no consulting services. Thus, we recommend for a totally in-house role engineering effort that no specialized tools be acquired initially. As experience is gained with the process, it will be possible at a later date to evaluate the cost-effectiveness of acquiring and using specialized tools. By implication, if specialized tools and consulting services are used, their estimated percentage of total role engineering costs will be 30% of the total, or one-half of the estimated in-house staffing cost.

No special justification for role engineering should be necessary if RBAC has been adequately justified. However, the scope of the role engineering effort will have to be considered as part of the justification process. This scope will determine potential benefits and costs.

While it is always good to be positive when approaching an upcoming effort, it can be valuable also to consider when it may not be advisable to go forward with the effort. A planned role engineering effort should be evaluated first as to where it will support the enterprise mission and objectives. To some extent this support will simultaneously be provided by security in general and then by RBAC in particular. However, a proposed role engineering effort must, independently from these, support the mission and objectives. This, of course, implies that the enterprise mission and objectives are known. When these are stated and acknowledged by the organization, they can be used as guidance. If they are not so, it will be necessary to obtain this information from management.

Given that a role engineering project does meet the organization's mission and objectives, what additional criteria can we use to identify projects that are advisable to carry out and those which should be either rejected or modified? Table 3.1 provides some of the criteria for judging potential role engineering projects.

The information provided in Chapter 12, Planning a Role Engineering Effort, includes a set of baseline rates of progress relating to a role engineering effort. For each goal, a unit of measure and a typical value or range are provided. The typical values or ranges can be used to estimate the level of effort that will be needed for a planned effort.

Table 3.1
Criteria for Judging Potential Role Engineering Projects

Positive Criteria. The Proposed Project Should:	**Negative Criteria. The Proposed Project Should Not:**
Use an established process	Require complex and/or costly tools that have not been previously used by the team
Be under a charter approved by management	Attempt to define all roles or solve all security management problems
Have clearly stated goals and timed milestones	Be conducted solely by technical personnel
Have a finite completion date	Be a research project
Have a completion date that coincides with related dependent projects	Push the state of the art of technology
Have an acceptable risk regarding on-time completion	Require participation of excessive numbers of subject matter experts
Have acceptable cost and staff estimates	Involve participation by representatives of an excessive number of organizational elements
Include experienced in-house staff or outside consultants	Be minimally staffed
Have numerous interim results	Be overstaffed
Capitalize on available sources of information and insights	Have several leaders
Have an identifiable source of funding	Have an excessive number of subgroups
Have an identified modus operandi (teleconferences, face-to-face meetings, collaboration tools)	Be dependent on one or two individuals

Acquisition/Development Phase

In the acquisition/development phase a functional statement of need will typically be established as a first step. As RBAC implementation entails the availability of enterprise role definitions, a role engineering effort will be needed if roles have not been already adequately engineered. An activity to investigate any existing role definitions that may be applicable should be considered (see Chapter 6). It is imperative to investigate in the planning process the availability of tools for establishing an RBAC environment, and it may also be advisable to conduct market research to investigate the availability of relevant tools for role engineering itself (see Chapter 9). Tools for establishing an RBAC environment are those IT infrastructure components that maintain role definitions in a manner whereby they are available for authorizing roles to users, evaluating RBAC policy statements, making access control decisions, and rendering these decisions to consuming applications. These infrastructure components should provide—perhaps in a modular fashion—support for sophisticated security policies such as separation of duties and other constraints. Role engineering tools are those used to assist in defining and maintaining roles with their various components. A role repository is an example of a tool that overlaps these two categories of tools.

In the acquisition/development phase the following set of activities should take place. While the primary focus in this phase is on defining, designing, and building an application, there is also some applicability to the role engineering effort. Table 3.2 summarizes the activities in the acquisition/development phase and indicates how role engineering activities would fit into the mix of activities.

Implementation Phase

In the implementation phase the following activities should take place.

Role Definition

Here is where the role engineering process is followed to define first high-level permissions and associated role names and then constraints and hierarchies. To complete the role definitions, system developers are engaged to translate the high level permissions into IT permissions.

Modeling and Testing

As roles are developed it is necessary to model and test them to ensure that they meet the access control policy. Modeling consists of creating sets of users, roles, and resources. Testing consists of ensuring that the roles are enforcing the access control policy.

Table 3.2
Acquisition/Development Activities

Activity	**Role Engineering Note**
Feasibility Study	**Required**
Estimate of work to be done	Scope—domain areas, systems, role granularity
Estimate of available resources	Staff (technical, administrative, domain expert), funds, office space, tools
Estimate of time to complete a usable set of role definitions	Based on scope, rate of progress, available staff
Requirements Analysis	**Recommended**
Identify domain areas for role definitions	Administrative, mission support, financial, technical
Types of roles needed	Structural, functional
Hierarchies needed (if any)	Related to scope
Constraints needed (if any)	Based on previous work, if available
Cost-Benefit Analysis	**Recommended**
Justification of requirements identified	Benefits should outweigh costs
Estimated development or acquisition costs	Staff; physical plant; purchase and tailoring of analysis tools, collaboration tools, repository
Identification and evaluation of anticipated benefits	Improved job descriptions, business streamlining, permission streamlining and documentation
Conversion Study	**Optional**
Identification of as-is access control scheme	Access control software, applications, middleware, protected objects, subject granularity
Identification of to-be access control scheme	Policy decision points, policy enforcement points, permissions, identity and access management
Process steps required	Definition tables, processing scripts, verification techniques
Pilot conversion subset	Domain areas, systems, organizations
Identification of conversion alternatives	Replacement, modification, consolidation
Development required after conversion	User interfaces, training materials, maintenance tools
Estimate level of effort	Staff, time frame, workload

Table 3.2 (continued)

Cost Analysis	**Recommended**
Develop work breakdown structure	Work packages
Estimate staffing needed	Support to complete work packages
Estimate staffing costs	Rates, hours, contracting
Estimate support costs	Materials, IT support, administrative
Risk Management Plan	**Optional**
Establish risk categories	Cost, schedule, results
Establish risk factors	Technology, expertise, impacts, dependencies
Conduct qualitative risk analysis	Ranking of risks based on likelihood and impact
Identify risk mitigation strategies	Avoidance, acceptance, scrutiny, rescoping
Role Engineering Planning	**Required**
Staffing	Types, numbers, phasing
Facilities	Office space, IT support,
Methods	Process, analysis techniques, naming conventions, cataloging techniques
Support tools	Current, new, licenses, training
Reviews	Experts, travel, rework effort
Publishing	Web-based, documents, repositories, media
Evaluation	Measures definition, collection, analysis, and reporting

Source: [2].

Initial User Training

Where new or modified user interfaces or procedures have been established as a result of the RBAC rollout, users will typically require orientation and training to make effective use of the new capabilities.

Documentation

Online, system embedded, or off-line documentation of new procedures and access methods will be required to complete the implementation of RBAC using the newly engineered roles.

Figure 3.2 illustrates the activities that would take place in the implementation phase.

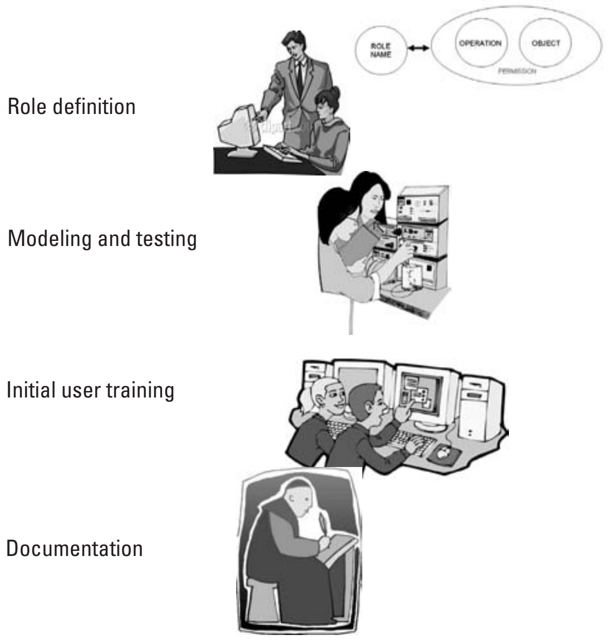


Figure 3.2 Activities that take place in the implementation phase. (Clip art copyright clipart.com.)

Operations and Maintenance Phase

In the operations and maintenance phase the following activities should take place.

Performance Measurement

Ongoing collection, analysis, and reporting of measures of efficiency and effectiveness will be conducted. Lessons learned will be fed back into the development and maintenance processes.

Operations

For role engineering, operations will consist of those activities needed to maintain role definitions, assess the effectiveness of support for the access control policy, and reengineering role definitions.

Maintenance

Maintenance is closely related to operations and covers the implementation of the changes arising as a result of role reengineering. This implementation consists of recording new definitions and reflecting those definitions in the systems to which they pertain.

Figure 3.3 illustrates the activities that would take place in the operations and maintenance phase.

Disposition Phase

In the disposition phase a conversion planning activity should take place. As a system or part of a system is being phased out, most likely a new system will take the place of the former one. Unless the protected objects and the access control policy will remain unchanged, the role structure will probably need to be reengineered to reflect the differences. Whether or not this

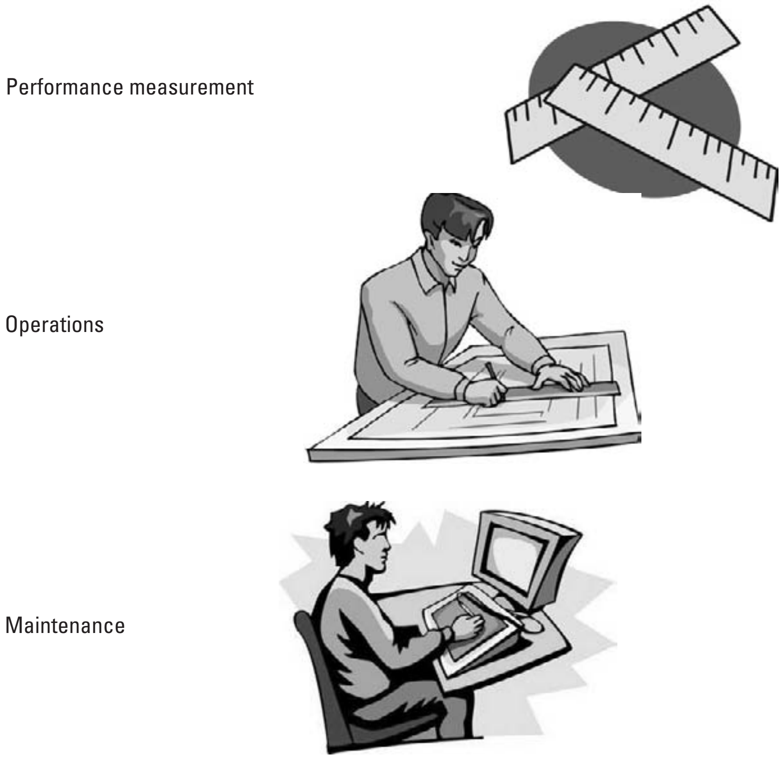


Figure 3.3 Activities that take place in the operations and maintenance phase. (Clip art copyright clipart.com.)

reengineering is needed will depend on other factors such as those addressed in the acquisition/development phase.

Figure 3.4 illustrates the activities that would take place in the disposition phase.

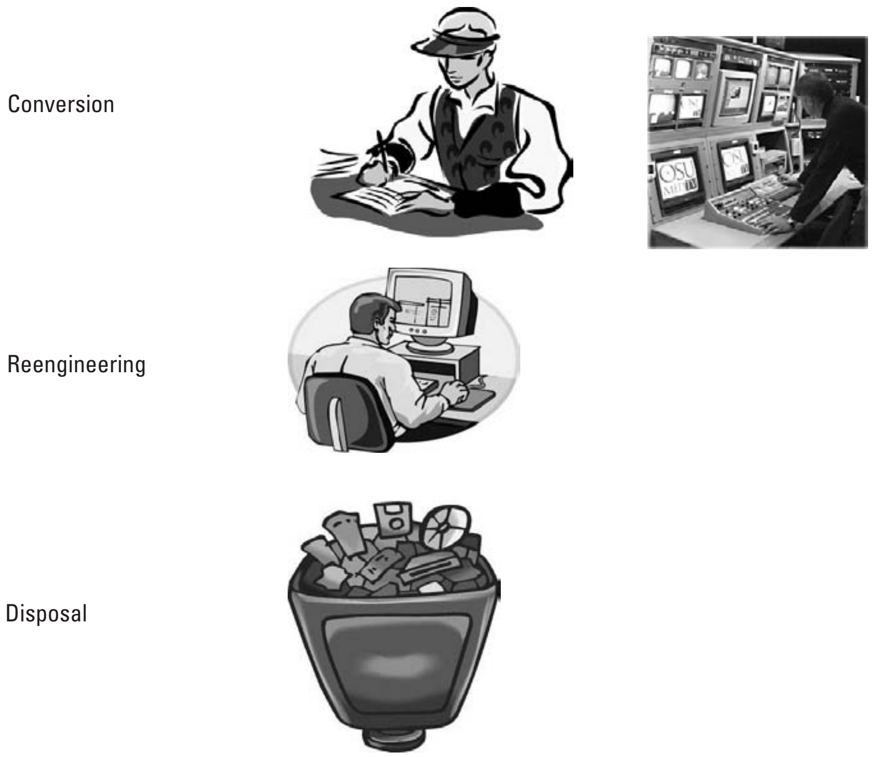


Figure 3.4 Activities that take place in the disposition phase. (Clip art copyright clipart.com.)

References

[1] Schimpf, G., "Role-Engineering—Critical Success Factors for Enterprise Security Administration," *16th Annual Computer Security Applications Conference Issues 2000: Enterprise Security Management Special Workshop*, New Orleans, LA, December 11–15, 2000.

[2] National Institute of Standards and Technology, *Security Considerations in the Information System Development Life Cycle*, NIST Special Publication 800-64 REV.1, June 2004.

[3] Kern, A., et al., "Observations on the Role Life-Cycle in the Context of Enterprise Security Management," *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, Monterey, CA, June 3–4, 2002, pp. 43–51.

[4] Cleland, D. I., and D. F. Kocaoglu, *Engineering Management*, New York: McGraw-Hill, 1981.

4

Role Engineering and Why We Need It

Once a decision has been made to adopt RBAC, the need for role definitions becomes apparent. If an RBAC infrastructure is considered as a car, role definitions are the fuel for the car (see Figure 4.1). Extending this analogy a little, it is important to put the correct fuel into the car and it is also important to provide role definitions that are the best available.

What Is Role Engineering?

Role engineering is the definition of a set of roles consisting of role names and permissions. Permissions consist of operations on objects. Constraints, if any, must also be included. Finally, the roles must be structured into hierarchies if they are being used. All of these required and optional components must be defined and structured into a cohesive whole such that the access control policy is supported. As the set of roles, constraints, and hierarchies is defined it will be necessary to follow a process of definition, construction, testing, and iteration. For example, if a set of roles with constraints and hierarchies is constructed and then tested to verify its complete support of the access control policy, it may turn out that certain aspects of the policy are not being supported. At this juncture it may be necessary to modify one or more role names, permissions, constraints, and hierarchies. As these components are all interrelated, a process of modification of the components must be followed to arrive at a satisfactory overall role structure. This is the engineering

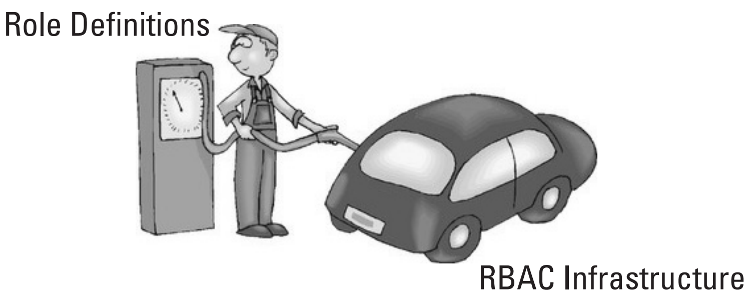
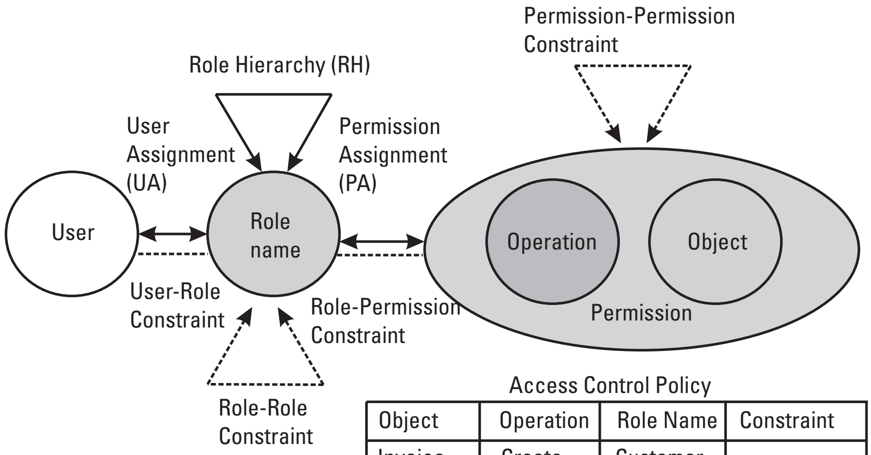


Figure 4.1 Illustration of gas (role definitions) going into a car (RBAC infrastructure). (Clip art copyright clipart.com.)

aspect of role engineering. Figure 4.2 illustrates the RBAC model and its relationship to an access control policy. In the figure the policy is represented as a table containing objects, operations, role names, and constraints. A more complete version of this policy table appears in Chapter 5.

What do we actually mean by role engineering? Surely, the role part of it is certain: we are trying to define roles to be used for RBAC. How about the engineering part? Because of the need to establish access control policy and then design a set of roles that effectively implements that policy, it is necessary to define the components of roles and then use the components to assemble roles, constraints, and hierarchies to create a role structure. There are many ways that this role structure can be realized. Developing the best role structure, given limitations of available engineering and subject matter personnel and resources, involves a process of design, trade-offs, and optimization. This is role engineering. Figure 4.3 illustrates the sample access control policy by featuring the role names that have assigned permissions in common with other role names. These role names are candidates for being in hierarchical relationships where one of the role names would inherit the common permissions from the other role names. Figure 4.4 illustrates those role names with permissions in common with other role names. The common permissions are shown in gray.

Observing the role names and their assigned permissions in Figure 4.4, we note that sets of two role names share permissions. These would be candidates for members of a common hierarchy where one role name inherits a shared permission from another role. Figure 4.5 illustrates an inheritance relationship where the access control policy is not adequately or correctly supported. Figure 4.6 illustrates an inheritance relationship where the access control policy is adequately supported. This is discussed in the next section.



Access Control Policy

Object	Operation	Role Name	Constraint
Invoice	Create	Customer	
Invoice	Display	Customer-service rep.	
Invoice	Display	Customer	
Invoice	Submit	System admin	ME: Payables specialist

Figure 4.2 Illustration of role model shown as components to be identified and engineered into a role structure that supports the access control policy.

An Example of Incorrect Engineering

In Figure 4.5, noting that customer-service-rep and payables-specialist roles share some permissions, a hierarchy was formed with customer-service-rep inheriting permissions from payables-specialist. The display payment permission was removed from customer-service-rep since that permission is assigned to payables-specialist and customer-service-rep will inherit that permission through the hierarchy. However, this role engineering choice does not result in adequate support of the access control policy as it gives some excess permissions to the customer-service-rep role. That is, customer-service-rep also inherits receive payment and post payment from payables-specialist and these assignments to customer-service-rep are not present in the access control policy.

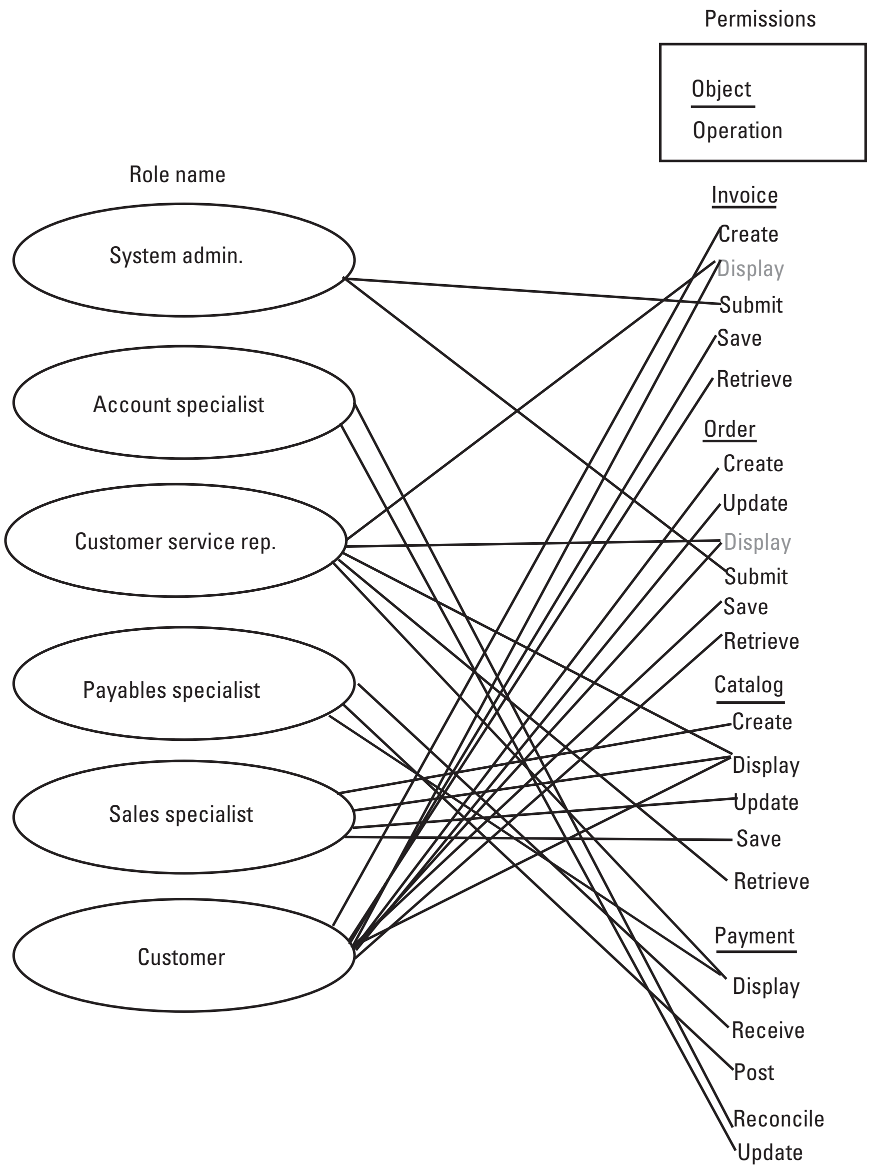


Figure 4.3 Role names with assigned permissions. Permissions common to two or more role names are shown in shading.

Candidates for hierarchical relationships

- Two or more role names have permissions in common.
- These role names MAY form a subhierarchy that supports the access control policy.
- Common permissions are shown in gray in this set of diagrams.

(a)

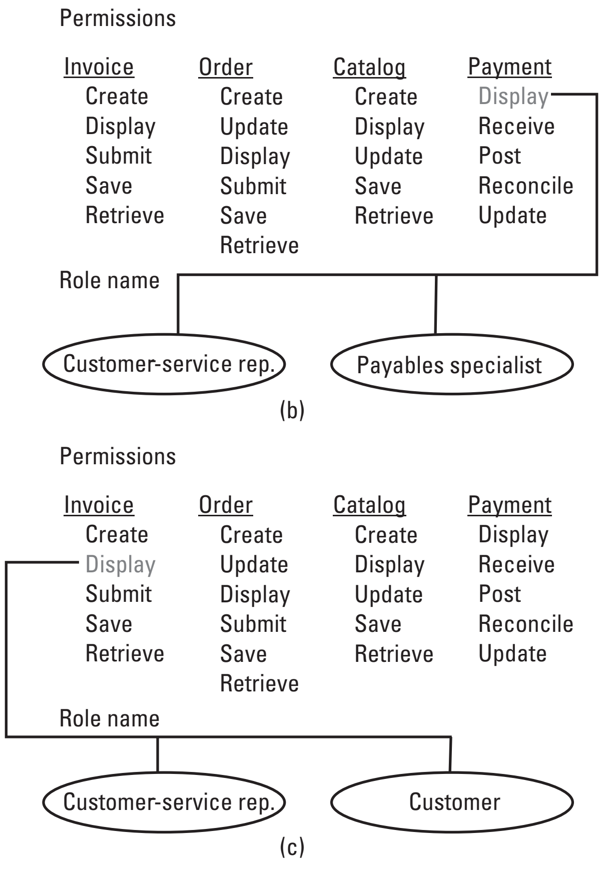
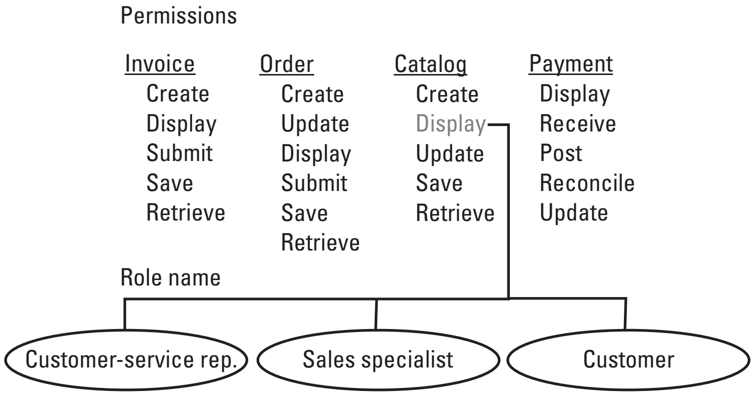
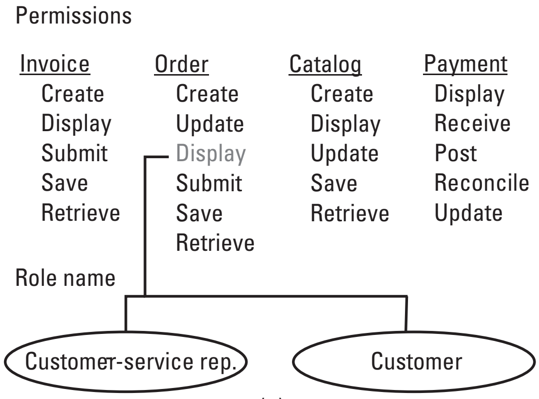


Figure 4.4 (a–h) Role names with permissions in common.

In Figure 4.6 noting that the customer and customer-service-rep share some permissions, a hierarchy was formed with customer inheriting permissions from customer-service-rep. The display invoice, display payment, display order, display catalog, and retrieve catalog permissions were removed from customer since those permissions are assigned to customer-service-rep



(d)



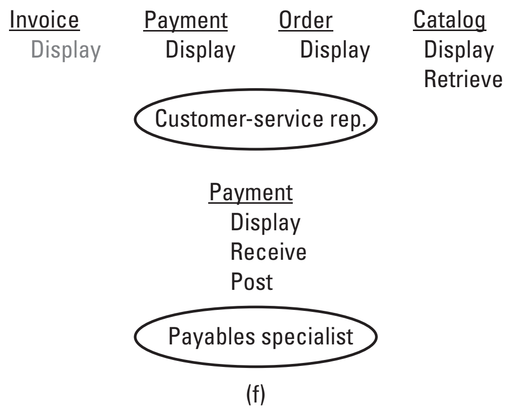
(e)

Figure 4.4 (continued)

and customer will inherit them through the hierarchy. In this case the access control policy is adequately supported because the permissions effectively assigned to the two role names match the policy exactly.

Sources of Roles

In our experience, many organizations embarking on the implementation of RBAC ask, logically, what should they use as a source of roles. In the healthcare field some materials are available as sources for role definitions. For example, the ASTM list of role names for healthcare [1] can be used as a



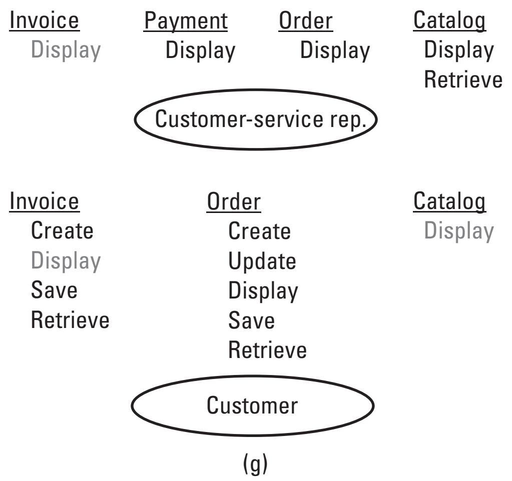


Figure 4.4 (continued)

starter set for roles. These were used by the Veterans Health Administration (VHA) as a type of scaffold to assist in the definition of standard healthcare permissions. The standard set of healthcare permissions [2] developed by VHA and the Health Level 7 (HL7) Security Technical Committee is now available as source material in developing roles. Of course, these materials can only be used in healthcare, and even in healthcare, there will be gaps in the availability of role names and permissions. Given this situation, how does an organization obtain the necessary permissions and role names? Obtaining

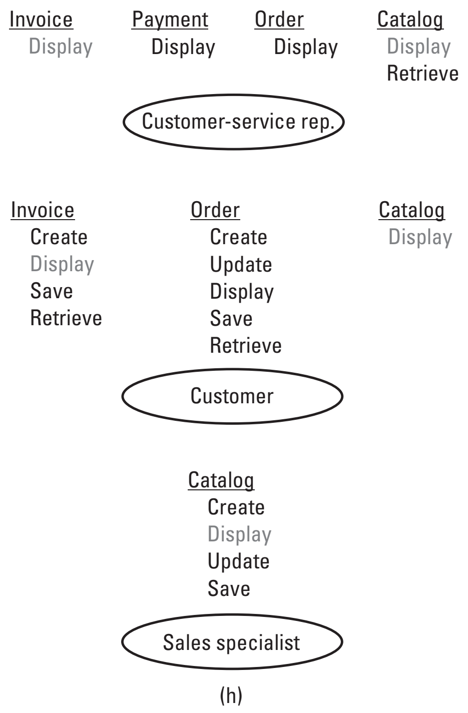


Figure 4.4 (continued)

these items is part of role engineering. Either a bottom-up or top-down approach can be taken to obtain these items from the organization. If results of prior efforts are available, their suitability should be assessed and a determination should be made whether or not to adopt existing results in the organization's role engineering effort. Figure 4.7 illustrates sources of role materials. Some sources can provide either role names or permissions separately, and possibly both. Other sources can provide both role names and permissions jointly. The sources consist of top-down process, bottom-up analysis of existing systems, human resources department, existing roles, access control policy, and standards.

Access Control Policy

As mentioned, role engineering must be carried out in the context of the organization's access control policy. This recognizes the "access control"

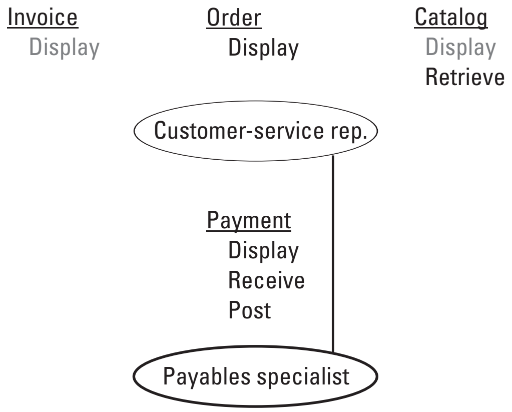


Figure 4.5 Illustration of role components that have been incorrectly engineered.

aspect of RBAC. Thus, a prerequisite of role engineering is the establishment of the access control policy that RBAC is to support.

Basic to that policy is a determination of what objects have to be secured and what operations are to be defined as permitted on those objects. These are used to compose permissions. Another fundamental requirement is to determine which job functions are security relevant. These are used to define role names. While the security-relevant job functions will be reflected in the access control policy, and therefore are the most critical, it is also advisable to identify any other job functions. These nonsecurity relevant job functions may be realized as roles that are given default permissions, and they may become security relevant at a later point in time. Figure 4.8 illustrates the components of access control policy, role structure, testing, and iteration.

Role Names and Permissions

To define role names, it is necessary to understand the intended differences in access control permissions from one role to another. If two roles will have the same set of permissions assigned to them, it may not make sense to have two distinct roles. Of course, if it is anticipated that at some time in the future the two roles' permissions may differ, the two role definitions may be advantageous. Similarly, if two roles with the same permission sets will be assigned to two different types of users, perhaps the two roles should be maintained with different role names. Figure 4.9 illustrates the sources that affect the selection of role names.

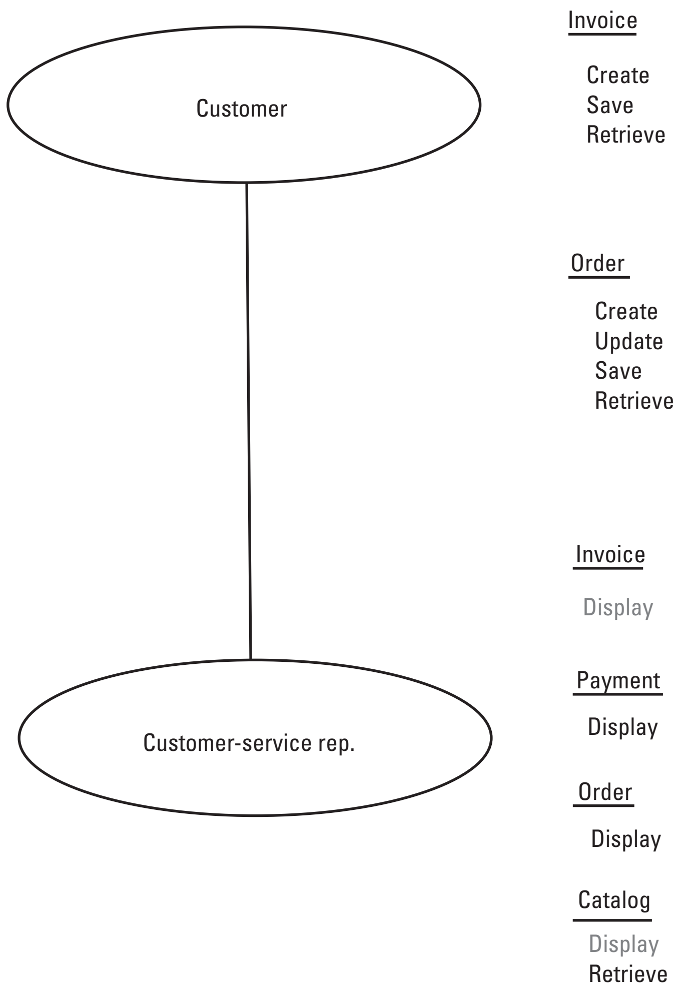


Figure 4.6 Illustration of role components that have been correctly engineered.

The role names are important in the assignment of users to roles. The permissions to be assigned to roles must also be known and understood before roles can be defined. As stated previously, the permissions form part of the organization's access control policy. Perhaps some permissions are already known at the outset of the role engineering effort. If so, these may be used as a starter set for a permission catalog. If these existing permissions are actually embedded within IT systems, it would be necessary to analyze these systems to extract the permissions from them. This would constitute a

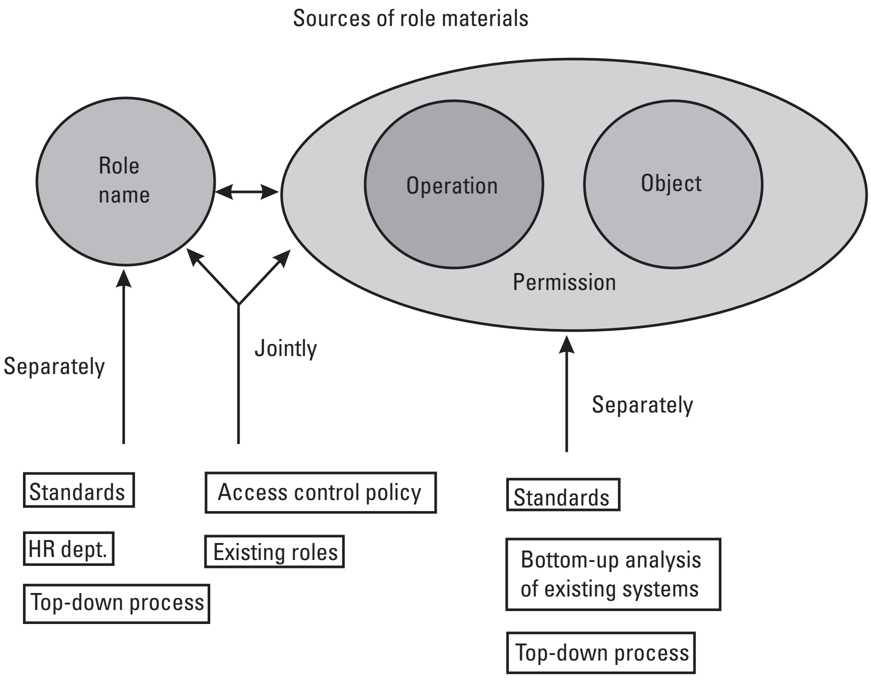
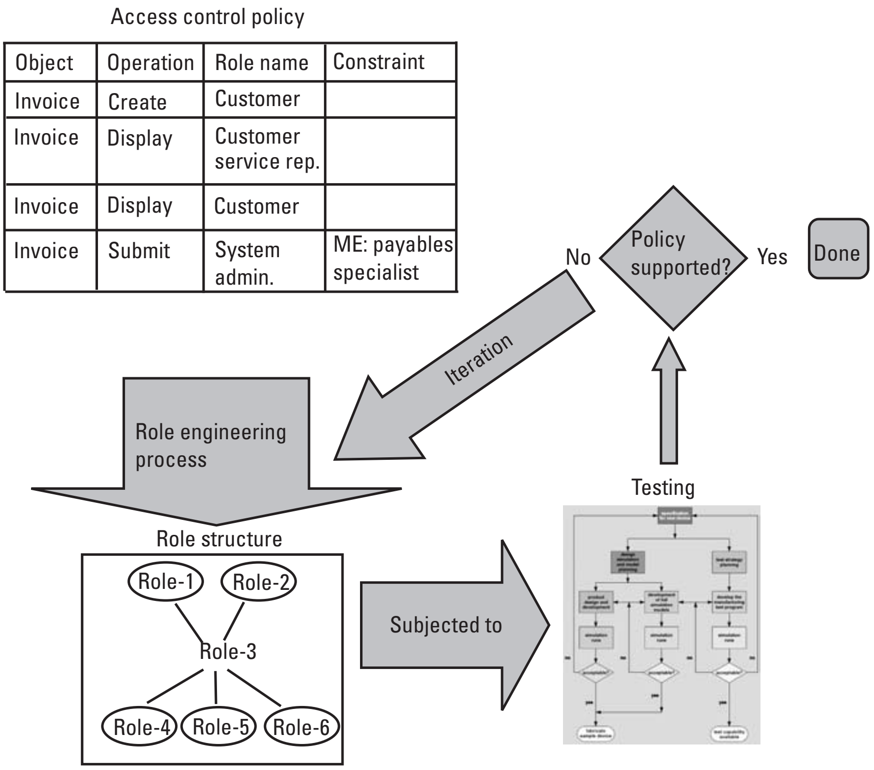


Figure 4.7 Diagram of sources of role materials: access control policy, standards, HR department, existing roles, bottom-up analysis of existing systems, top-down process.

bottom-up approach to role engineering. A top-down approach to role definition will probably also be warranted. Figure 4.10 illustrates the relationship between access control policy and permission definitions.

Non-RBAC Support of the Access Control Policy

The access control policy may not be entirely supported by RBAC. One example of this would be where access permissions must be assigned to an individual by name, perhaps because this person uniquely possesses a restricted authorization. In this case, the security policy would include the direct assignment of the permission to the designated individual. This is an example of entity-based access control. Another example would be where different parts of the system use different technologies for security, and where the application of RBAC does not readily lend itself to supporting the access control policy. For example, if Microsoft Windows is being used to



Access control policy

Object	Operation	Role name	Constraint
Invoice	Create	Customer	
Invoice	Display	Customer service rep.	
Invoice	Display	Customer	
Invoice	Submit	System admin.	ME: payables specialist

Figure 4.8 Illustration of access control policy, role structure, testing, and iteration.

authenticate users and to control their access to certain system resources, such as applications or printers, it may be desirable to maintain such an entity-based access control scheme for the Microsoft Windows environment and to reserve RBAC for the functional access within applications. In this situation, the Microsoft Windows environment would be effectively controlling permissions that would typically be assigned to structural, rather than functional, roles. Figure 4.11 illustrates the access control possibilities to support the access control policy.

Some or all users can receive access to resources via roles. We have mentioned one case where assignment of a permission is not made through a role, where a distinguished individual may merit unique and direct assignment of permissions. Further, since RBAC implies the ownership of protected

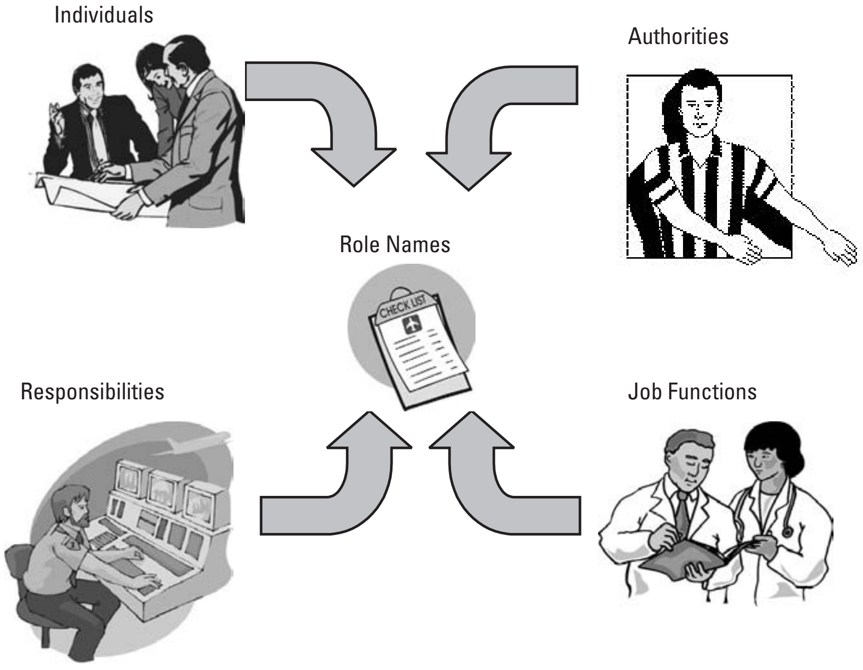
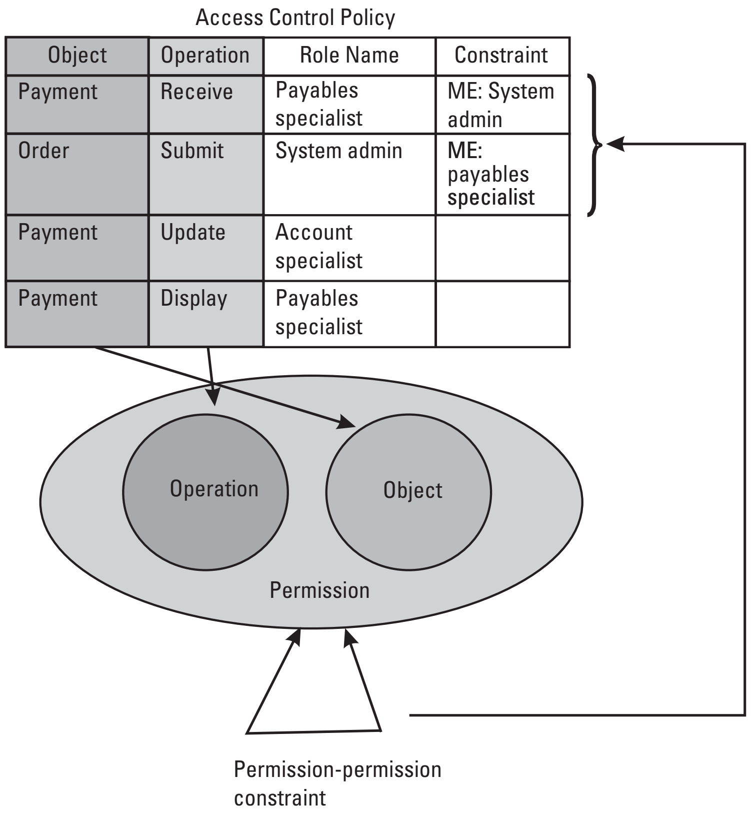


Figure 4.9 Illustration of individuals, responsibilities, authorities, and job functions, leading to role names. (Clip art copyright clipart.com.)

resources at the organizational level, it is a mechanism or construct to implement mandatory access control policies. However, not all resources within an IT system are owned by the organization. For example, personal or preliminary files may be owned and used by individuals in their own or shared address space. These individually owned resources may be placed outside the RBAC-controlled regime if desired.

Resources Subject to RBAC

Another prerequisite to role engineering is to determine which systems or other IT resources will be accessed using RBAC. For practical as well as policy reasons, it may not be advisable to use RBAC for a particular resource. For example, a legacy system whose lifetime is limited would likely not be a candidate for using an RBAC approach. Nor would a system that cannot



Access Control Policy

Object	Operation	Role Name	Constraint
Payment	Receive	Payables specialist	ME: System admin
Order	Submit	System admin	ME: payables specialist
Payment	Update	Account specialist	
Payment	Display	Payables specialist	

Figure 4.10 Illustration of relationship between access control policy and permissions.

be easily adapted to using an external policy decision point be a likely candidate.

Constraints

The access control policy may include constraints. These may be between permissions, between permissions and role names, between roles, and between users and roles. While constraints in the policy should be identified as early in the process as possible, in practice it is possible to add constraints to the RBAC mechanisms after the initial role structure has been defined and designed. Figure 4.12 illustrates constraints among users, roles, and permissions.

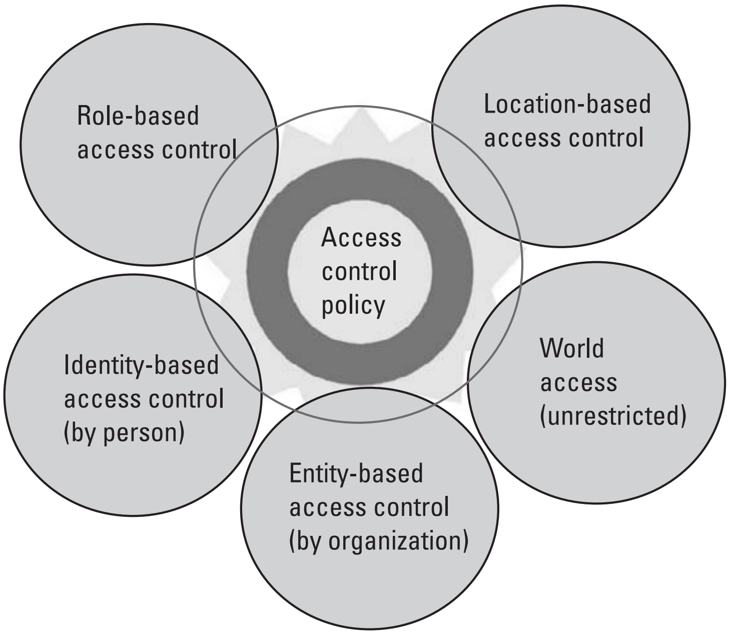

Role-based access control
Location-based access control
Access control policy
Identity-based access control (by person)
World access (unrestricted)
Entity-based access control (by organization)

Figure 4.11 Illustration of access control policy that is supported by RBAC and other than RBAC.

Use of Hierarchies

Whether or not to use hierarchies is not highly dependent on the access control policy, but the use of hierarchies should be considered early on in the role engineering process. This is because the roles in a hierarchy function as a set of roles as a whole and in part—that is, each role in the hierarchy can be considered as an available role to which a user can be assigned. In the role engineering process, any role in the hierarchy can be subdivided as to its permissions, and the permissions removed from the role can be assigned to roles lower in the hierarchy. Then the senior role can inherit the removed permissions from the junior roles. Figure 4.13 illustrates the process of creating junior roles from a senior role, with removal of permissions from a senior role and placement on a junior role.

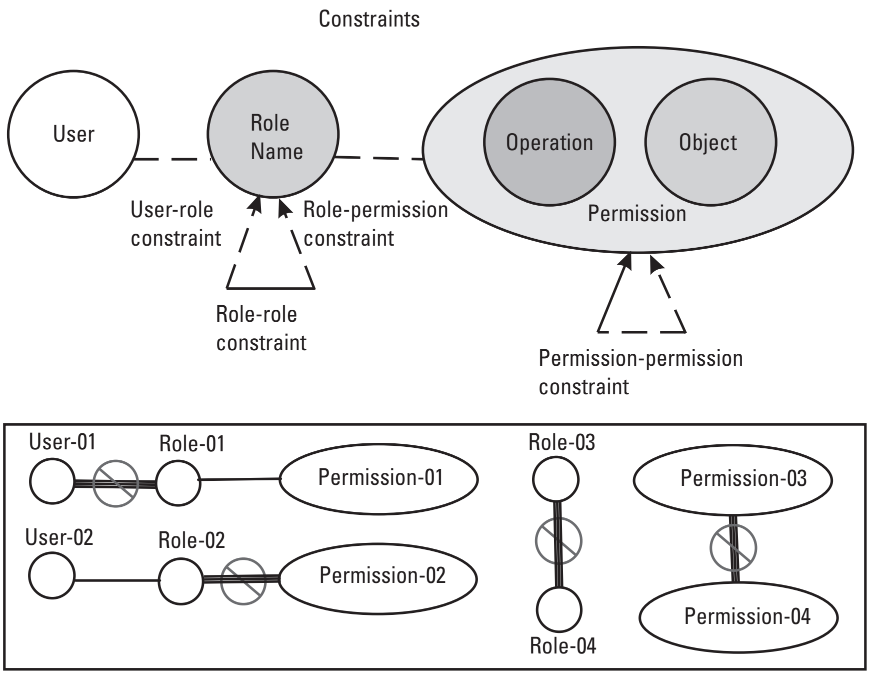


Figure 4.12 Constraints between permission-permission, permission-role, role-role, and user-role.

Realization of Roles in IT Systems

Roles may be envisioned in several ways. Possibly the most intuitive way is in accordance with the models in the RBAC standard. Here are defined objects called Role, Permission, Operation, and Object. This is the value of the models. However, in an IT system it may or may not be the case that these objects are present. For example, a system can follow the RBAC standard without having something in the system called a role. Thus, the adoption of RBAC does not imply that a system will be designed as an implementation of the models in the RBAC standard. The models are more conceptual in nature rather than being objects that must be reflected in a system design. In practice, for many RBAC systems there will be a traceability of the system design to the RBAC models but not a one-to-one correspondence to them.

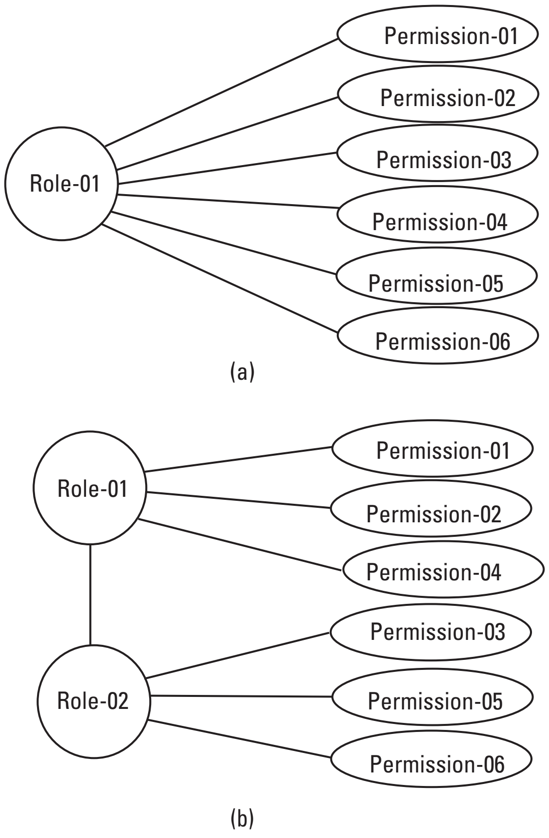


Figure 4.13 Process of creating a junior role from a senior role. (a) Senior role containing a set of permissions. (b) Removal of permissions from a senior role and placement on a junior role.

Figure 4.14 illustrates some IT components and functions that may implement the RBAC model.

Similarly, the permissions within an IT system do not have to be recognizable as operations on objects. However, the high-level permissions must be defined as operation-object pairs. Policy enforcement points [3] exist within applications and other system components that enforce access control decisions. Since the policy enforcement points in a system exist in a variety of forms as software functions and access control information, it is actually unlikely that the permissions would be recognizable as operation-object pairs at this detailed level.

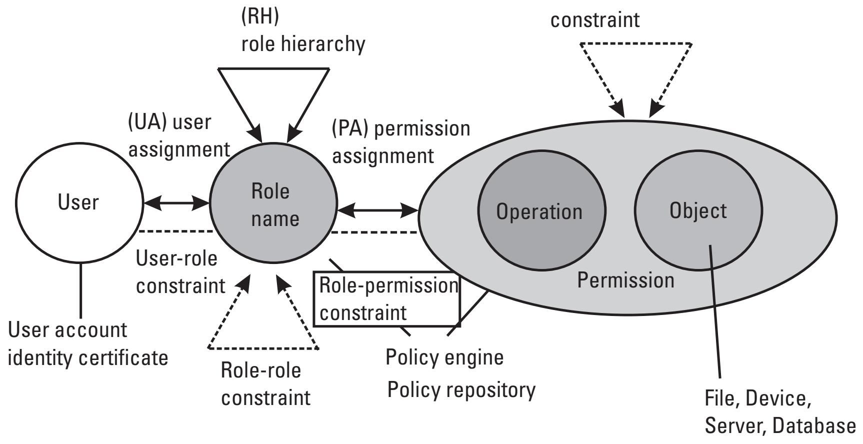


Figure 4.14 RBAC model realized as IT components and functions.

RBAC is more a means of supporting access control policy than a mechanism to be used in policy definition and enforcement. Once roles and permissions have been defined, their realization in systems is a matter of design. As stated earlier, the RBAC models do not necessarily have to be used in the system design. Employment of RBAC can benefit from infrastructure products that facilitate the establishment of an RBAC environment. However, RBAC does not depend on using these products. For example, sometimes roles can be implemented as user profiles or groups attached to files or other resources. Whether or not to use specialized RBAC products or to use other available mechanisms, such as operating systems and databases, is a business decision. Whatever mechanism(s) is selected to implement RBAC, it is recommended to establish a user interface that permits an administrative person to associate a user with a role. Underlying that interface can be any number of tools and infrastructure support.

We point out some of the variability in RBAC implementations. This variability reflects how roles can be used for access control in IT systems. For example, some or all information resources may be protected by roles. This depends on access control policy and practical considerations. Ideally, for consistency and transparency, all access control would be under an RBAC regime. Variations from this ideal will involve an analysis of trade-offs to assess the alternatives.

Structural Roles and Functional Roles

It is useful to distinguish between structural roles and functional roles [4]. In this definition, structural roles have permissions that simply permit access to top-level resources (i.e., to resources at their entry points). A structural role might control access to an application's entry point, whereby a user could only open that application if he or she had been assigned to a structural role that grants that access. For example, before a user could enter a patient record system, the user would have to be assigned to a structural role that has that application as a permission. Functional roles are defined as controlling access to resources within applications. Both types of roles are used to define and enforce the access control policy. The role engineering process includes definition of both structural roles and functional roles. Since structural roles tend to be simpler that functional roles, it is advisable to define structural roles before defining functional roles. This approach can provide some RBAC advantages early on in the role engineering process. Figure 4.15 presents models of structural and functional roles. In the figure, both basic and functional roles involve an access control decision function (ADF) and an access control enforcement function (AEF). These are also known as a policy decision point (PDP) and a policy enforcement point (PEP), respectively. The major difference between the structural role and the functional roles is in the location of the AEF. While in both types of role the ADF is outside the target, for a structural role the AEF is outside the target and for a functional role the AEF is within the target. Figure 4.15 illustrates structural roles and functional roles.

Role Engineering as Requirements Engineering

Role engineering is similar to other systems development activities. It falls into the category of requirements engineering. Traditionally, requirements are statements of objectives and specifications. In role engineering, the requirements are specifications representing role names, permissions, constraints, and hierarchies.

Interviewing users to identify needs is important to role engineering, as it is in collecting other types of requirements. Subject matter experts must provide input to role engineering, as they possess the domain knowledge and experience that must be reflected in the role specifications. Individual interviews may be conducted to elicit the requirements or facilitated working

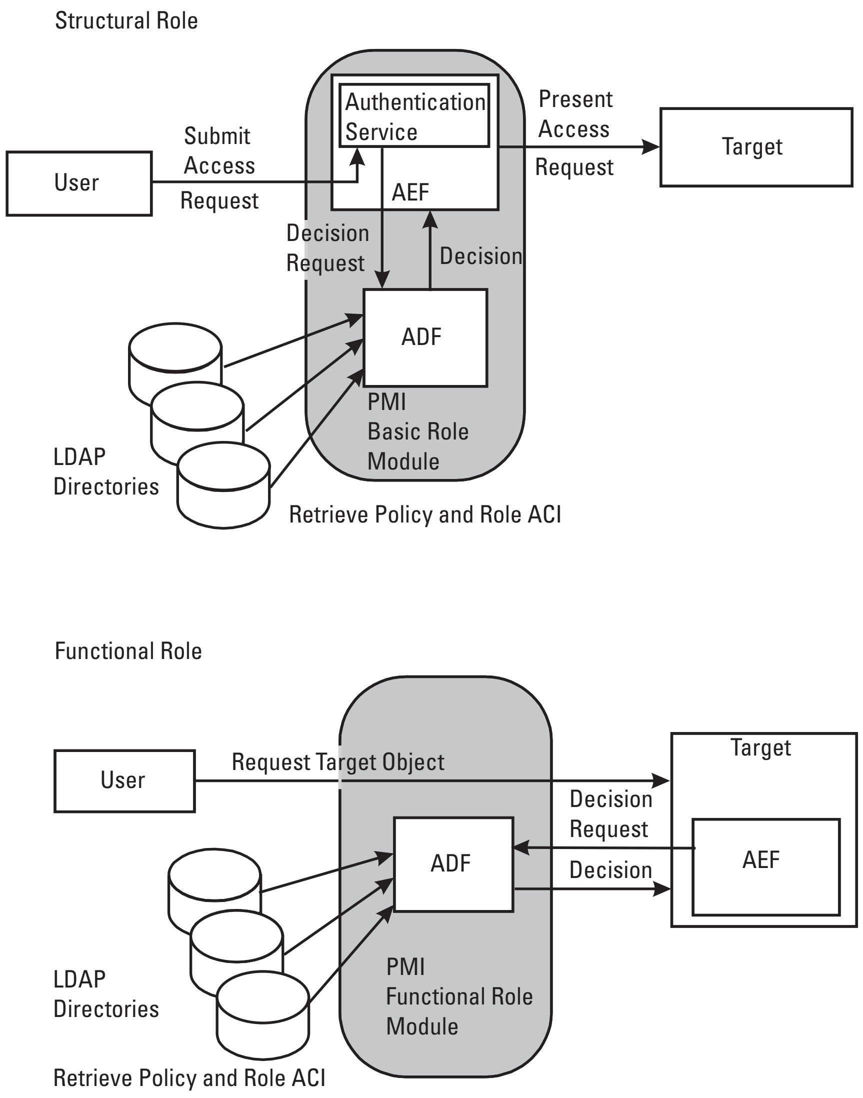


Figure 4.15 Structural roles and functional roles. LDAP: Lightweight Directory Access Protocol; PMI: privilege management infrastructure; ACI: access control information; ADF: access control decision function; AEF: access control enforcement function. (*After:* [5].)

sessions with subject matter experts may be used. Figure 4.16 illustrates the refinement from system-level permission requirements, such as those identified by subject matter experts, to IT requirements suitable for use by system developers. Actual system objects and code will be determined during the system design process and will vary from project to project.

Role Engineering as Systems Engineering

Creating and testing hypotheses regarding user requirements is applicable to role engineering, as it is with other systems engineering activities. One of the roles of the interviewer or facilitator is to analyze input from subject matter experts and to attempt to characterize and generalize preliminary results. These tentative results, in turn, assist the subject matter experts in deciding on the best definitions of role components. Figure 4.17 illustrates the process of identifying, assembling, analyzing, sorting, and normalizing permissions. Figure 4.18 provides a sample permission catalog, the result of the permission identification and engineering process. In the sample, the column labeled Basic Permission Name is actually the IT permissions that were identified during the illustrated permission engineering process.

Additionally, applying technical, administrative, and legal constraints to requirements is applicable to role engineering as well as to other forms of requirements engineering.

System requirements

Scenario ID	Permission ID	Requirement
SOE-002	POE-001	New laboratory order
SOE-002	POE-002	Change/discontinue laboratory order
SOE-001	POE-003	New radiology order
SOE-007	POE-004	Change/discontinue radiology order
SOE-001	POE-005	New/renew outpatient prescription order

IT operations on objects

POE-001	{C, laboratory order}
POE-002	{U, laboratory order}
POE-003	{C, radiology order}
POE-004	{U, radiology order}
POE-005	{C, outpatient prescription order}

Figure 4.16 Refinement of permission requirements, from system requirements to IT operations on objects.

One of the roles of the interviewer or facilitator is to analyze input from subject matter experts and subject it to the relevant external constraints that must be applied—this serves to guide the role definition process and to avoid possible rework when these types of constraints are applied too late

Translating requirements into systems artifacts is applicable to role engineering as well. Once role definitions have been established, they will be used in systems. This entails working with system developers to ensure roles are appropriately represented. A critical part of this work is interpreting permissions within the context of applications.

The process of interpreting permissions within the context of applications is actually no different than the process of realizing other high-level system requirements as implementation requirements through the creation of derived requirements from the system requirements. This process is the responsibility of system developers and therefore is outside the scope of role engineering per se. Still, personnel doing role engineering work must be aware that the permissions they define will be later translated into detailed system operations and parameters through a process of requirements derivation and translation. The role engineering personnel may be called upon during the system implementation phases of the system development life cycle to interpret high-level requirements for the edification of development personnel.

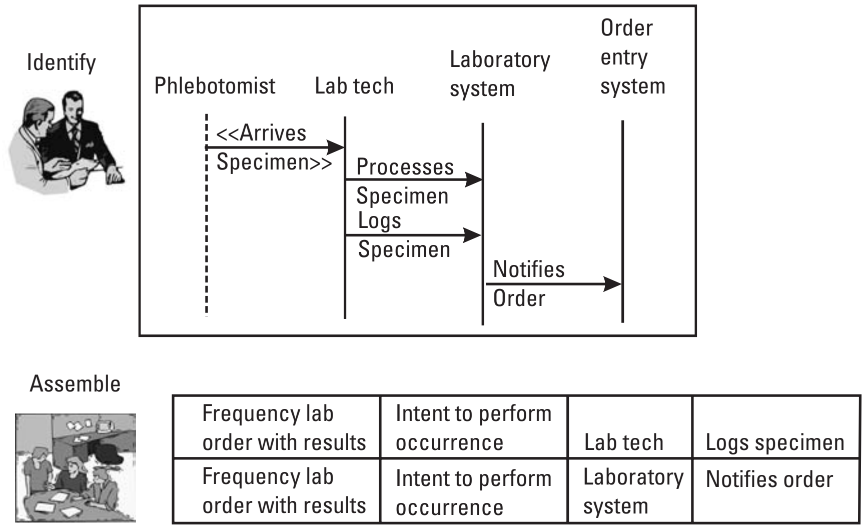


Figure 4.17 Process of identifying, assembling, analyzing, sorting, and normalizing permissions. (Clip art copyright clipart.com.)

Analyze

Scenario	Actor	Step	{OPERATION, OBJECT}
Intent to perform order—collect specimen	Phlebotomist	Receives STAT order	{R, Order]
Intent to perform order—collect specimen	Phlebotomist	Collects specimen	{C, Observation}, {U, Order}, {R, WorkList}
Intent to perform order—collect specimen	Phlebotomist	Prints STAT label	{C, Device}
Intent to perform order—collect specimen	Phlebotomist	Labels specimen	{C, Container}
Intent to perform order—collect specimen	Phlebotomist	Arrives specimen	{U, Observation}, {U, Order}
Intent to perform order—process specimen	Lab tech	Processes specimen	{U, Observation}
Intent to perform order—process specimen	Lab tech	Logs specimen	{U, Observation}
Intent to perform order—process specimen	Laboratory system	Notifies order	{U, Order}
Intent to perform occurrence	Phlebotomist	Collects specimen	{C, Observation}, {U, Order}, {R, WorkList}
Intent to perform occurrence	Phlebotomist	Arrives specimen	{U, Observation}, {U, Order}
Intent to perform occurrence	Lab tech	Processes specimen	{U, Observation}
Intent to perform occurrence	Lab tech	Logs specimen	{U, Observation}
Intent to perform occurrence	Laboratory system	Notifies order	{U, Order}

Figure 4.17 (continued)

Normalize

Scenario	Actor	Step	**{OPERATION, OBJECT}**
Intent to perform order—collect specimen	Phlebotomist	Receives STAT order	{R, Order}
Intent to perform order—collect specimen	Phlebotomist	Collects specimen	{C, Observation}, {U, Order}, {R, WorkList}
Intent to perform occurrence	Phlebotomist	Collects specimen	
Intent to perform order—collect specimen	Phlebotomist	Prints STAT label	{C, Device}
Intent to perform order—collect specimen	Phlebotomist	Labels specimen	{C, Container}
Intent to perform order—process specimen	Phlebotomist	Arrives specimen	{U, Observation}, {U, Order}
Intent to perform occurrence	Phlebotomist	Arrives specimen	
Intent to perform order—process specimen	Lab tech	Processes specimen	{U, Observation}
Intent to perform occurrence	Lab tech	Processes specimen	
Intent to perform order—process specimen	Lab tech	Logs specimen	{U, Observation}

Figure 4.17 (continued)

Scenario ID	Unique Permission ID	Abstract Permission Name	Basic Permission Name
Scen_1	Perm_1	Receives STAT order	{R, Order}
Scen_2	Perm_2	Collects specimen	{C, Observation}, {U, Order}, {R, Worklist}
Scen_3	Perm_3	Prints STAT label	{C, Device}
Scen_4	Perm_4	Labels specimen	{C, Container}
Scen_5	Perm_5	Arrives specimen	{U, Observation}, {U, Order}
Scen_6	Perm_6	Processes specimen	{U, Observation}
Scen_7	Perm_7	Logs specimen	{U, Observation}
Scen_8	Perm_8	Notifies order	{U, Order}

Figure 4.18 Sample permission catalog.

References

[1] *ASTM E 1986–98, Standard Guide for Information Access Privileges to Health Information,* approved October 10, 1998, November 1998.

[2] VHA-RBAC, http://www.va.gov/RBAC/documents.asp.

[3] OASIS XACML, http://www.oasis-open.org/committees/xacml/faq.php.

[4] International Standards Organization, Health Informatics—Functional and Structural Roles, Draft Standard for Comments, TC 215/WG4/N214, ISO/PDTS 21298, January 15, 2004.

[5] Chadwick, D. W., and A. Otenko, "The PERMIS X.509 Role Based Privilege Management Infrastructure," *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, Monterey, CA, June 3–4, 2002, pp. 135–140.

5

Defining Good Roles

What are good roles and what are bad roles? “Good” and “bad” in the sense of good or bad roles refers to the characteristics of the role definitions resulting from a role definition process. Good characteristics are those that make the roles easier to manage and also support the access control policy effectively. Bad characteristics are those that make roles more difficult to manage and that are questionable in their support of the access control policy. Figure 5.1 illustrates good and bad roles.

Among the advantages of RBAC are support for the security principle of least privilege and for the separation of duties. These must be considered in establishing access control policy and in defining roles accordingly. The principle of least privilege refers to the granting of only the minimal privileges needed by a given user to perform his or her job functions. Separation of duties refers to the restriction of a user from performing two operations that together could compromise the security of the system.

This chapter focuses on defining good roles, or at least roles that can be improved later. Good roles have the following characteristics:

- Readily recognizable names and clear patterns of constraint definitions and hierarchy relationships—these names could reflect the organization structure or the classes of users of the organization’s IT resources;

Bad Roles

Difficult to understand

Inadequately support access control policy

Object	Operation	Role Name	Constraint
Invoice	Create	Customer	
Invoice	Print	Representative	
Invoice	Display	Customer	
Invoice	Submit	Computer	

Good Roles

Easy to understand

Adequately support access control policy

Object	Operation	Role Name	Constraint
Invoice	Create	Customer	
Invoice	Display	Customer-service rep.	
Invoice	Display	Customer	
Invoice	Submit	System admin.	ME: payables specialist

Figure 5.1 Good and bad roles and the differences between them. (Clip art copyright clipart.com.)

- Permission sets that accurately represent the access control policy;
- Constraints that accurately represent the access control policy;
- Hierarchies that function correctly with permissions and constraints to accurately represent the access control policy.

Types of Roles

Several types of roles may be defined. For example, roles may be developed in various subject domains—these may include the areas of business, finance, administration, and security. Also, both structural and functional roles may be defined. Structural roles have permissions that permit connection or access to a gross level IT resource, such as a network, a server, a workflow, or a device. On the other hand, functional roles have permissions that determine what specific objects a user can access and perform operations on once a connection to a gross level resource has been made. Figure 5.2 illustrates roles in different domains as well as structural versus functional roles.

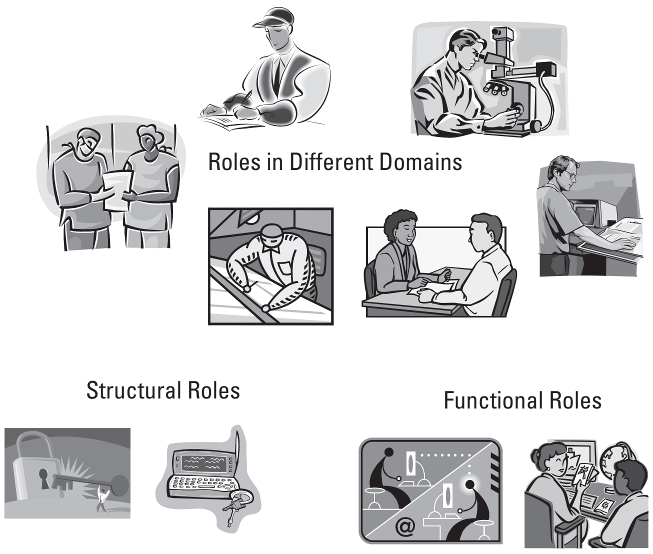
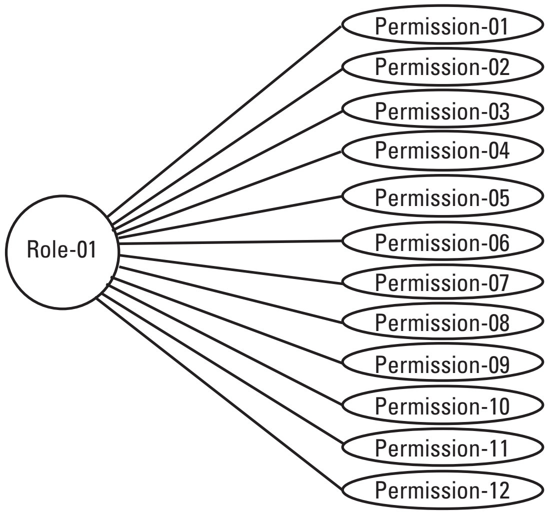


Figure 5.2 Roles in different domains as well as structural and functional roles. (Clip art copyright clipart.com.)

Role Engineering Guidelines

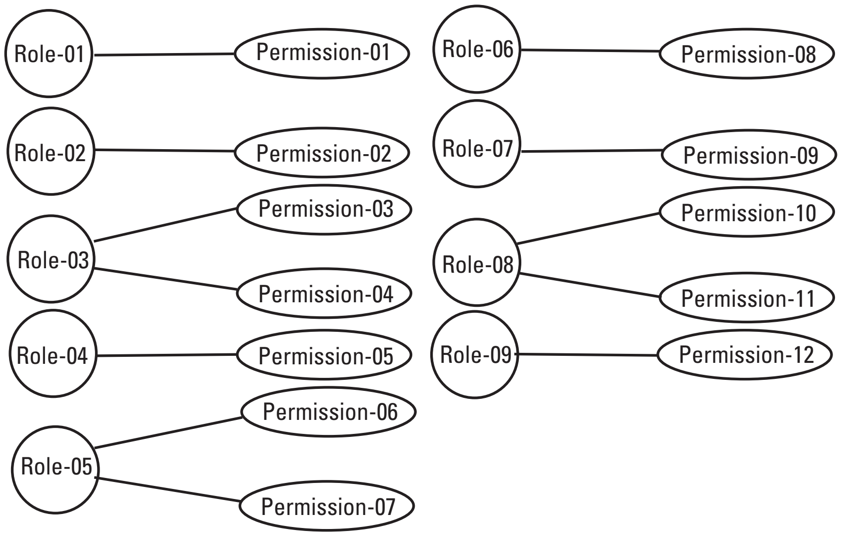
It is advisable to define structural roles first, as they are simpler than functional roles and may be subsequently enhanced to also be functional roles. The simplicity derives from the fact that permissions are simple "connect" permissions and the use of constraints and hierarchies is expected to be absent or limited.

Related to the quality of roles is the number of roles to be defined. In the extreme case where too few roles are defined, the roles will tend to have many permissions, with these permissions being, in many cases, unrelated semantically. Therefore, users assigned to these roles will tend to receive more permissions than they would need according to the principle of least privilege. In the other extreme case where too many roles are defined, users will tend to need many roles to perform their jobs. Also, the roles will tend to be more abstract than would be desired to permit administrative staff to understand the role structure and effectively assign or deassign users to roles. Thus, a balance must be struck between having too few roles and too many roles. Figure 5.3 illustrates a role with relatively too many permissions and



Possibility of failure to observe principle of least privilege

(a)



Possibility of confusion as to which roles need to be assigned to a given user

(b)

Figure 5.3 (a) A role with too many permissions (too few roles). (b) A large number of roles with fewer permissions (too many roles).

the case of a large number of roles with fewer permissions. We note that the illustrations are relative in nature, rather than depicting actual numbers of permissions on a role or the actual number of roles.

Criteria for achieving this balance are as follows:

- Role names should be readily recognizable by personnel assigning users to roles and focused on a particular job function.
- Permissions granted to each user via a role should enforce the principle of least privilege as well as separation of duties.
- The total number of roles should be considerably less than the number of users to be assigned to the roles; if the two numbers are comparable, the administrative advantage of RBAC is not being realized.

The approach to be taken in defining a set of roles should be to define the minimum essential number of roles and then to review these to see whether the roles with their permissions enforce the principle of least privilege. At this point the number of roles may need to be increased until the least privilege enforcement is being achieved overall. As the number of roles is increased, the average number of permissions assigned to the roles will be correspondingly decreased. Figure 5.4 illustrates the adjustment of the number of roles and permissions attached to roles from a situation with least privilege not being enforced compared with a situation with least privilege being enforced.

Access Control Policy

A prerequisite to defining a role structure is the existence of an access control policy to be enforced using RBAC. What would such a policy look like and how could it be defined?

Objects to Be Protected

For purposes of role engineering we need to define what objects in an information system or other IT resource are to be protected by the access control policy. Identification of these objects is fundamental to determining the access controls that are needed. Without objects to be protected, we have no

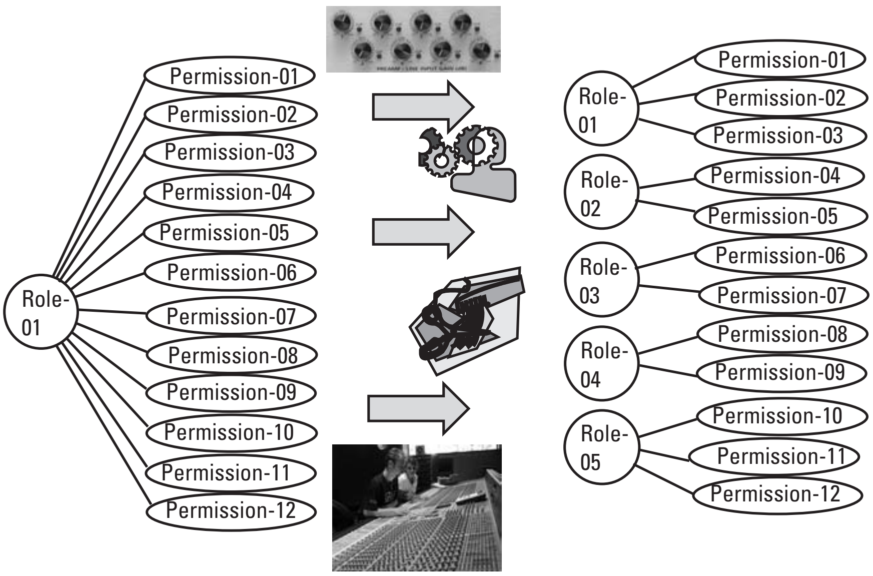


Figure 5.4 Adjustment of number of roles and permissions attached to roles. (Clip art copyright clipart.com.)

need for an access control policy. The particular objects to be protected are specific to each organization. Also specific are the modes of access to these objects. The rules governing these access modes will be based on such sources as business processes, laws and regulations, best practices, and the traditional security considerations of threats, vulnerabilities, and countermeasures. Figure 5.5 illustrates types of objects to be protected with sources of identification of these objects, as well as an indication of how objects are to be accessed.

How do we determine which objects are to be protected? In identifying the organization's protected objects, care must be taken to define the objects at the proper level of abstraction. This level will be suitable for inclusion of the object in access control rules. In defining objects for functional roles, data modeling and relational database concepts may be useful; for example, the objects can be entities which are realized as rows identified by primary keys. In defining objects for structural roles, these will typically be physical devices such as printers or logical devices such as servers.

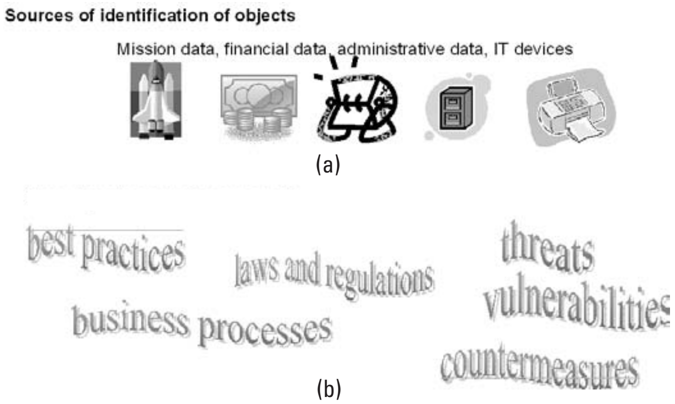

Sources of identification of objects
Mission data, financial data, administrative data, IT devices
(a)
best practices
laws and regulations
threats
vulnerabilities
business processes
countermeasures
(b)

Figure 5.5 Types of objects to be protected: (a) source of identification, and (b) how objects are to be accessed. (Clip art copyright clipart.com.)

Identifying Protected Objects

Sources of data for identifying the protected objects include existing data models, working applications and database designs, workflow definitions, forms, and objects referenced in existing statements of security policy. Just as the entities of a data model represent real-life objects, such as customers and shipments, the objects identified for purposes of an access control policy can and should represent real-life objects of the same nature.

Once the protected objects are identified, at least partially, the access modes pertinent to those objects must be identified. At this point we enter the definition of permissions. Permissions are defined by the RBAC standard to be operations on objects. As we discuss in Chapter 8, the operations on objects can be at any level of abstraction. For purposes of access control policy it is advantageous to define the operations and objects at a high level of abstraction, a level that results in operations and objects that are recognizable by non-IT specialists. Of course, these high-level definitions will need to be brought down to a detailed level, ultimately to the level of the IT system (or other resource) itself. But for the access control policy, it is much more efficient to define the policy in high-level terms. This will facilitate the verification of permissions against security policy and security-relevant requirements, and the operations on objects will be understandable by non-IT specialists. Figure 5.6 illustrates the high-level statement of permissions and realization of those permissions in successive layers of access control enforcement mechanisms.

Role Names

Once permissions are defined, we will need a set of role names. Here we are really outside the IT realm. We will reenter the IT realm when we start assigning permissions to role names. The role names must reflect the organization's job functions, personnel responsibilities, authorities, and other particulars of the organization. It will be preferable to define each role name and then assign appropriate permissions to the name. The alternative of defining all role names first, before assigning any permissions, would complicate the process. This is because the role name is in some ways related to the permissions, such that choosing a role name to some extent implies the selection of potential permissions for the role. So we define each role name and then select an appropriate set of permissions for the role. In this process it may be necessary to define additional permissions or to split existing permissions (e.g., separating out the operations on the objects). This is why it is advisable to define role names one by one. Figure 5.7 shows a set of role names defined

High-level permission	Write Prescription {Operation, Object}
Mid-level system components	Create Prescription Message {Operation, Object}
	Send Prescription Message {Operation, Object}
Lower-level system components	Read Prescription Message {Operation, Object}
	Open Medication Table {Operation, Object}
	Create Prescription Table {Operation, Object}
	Create Pharmacy Message {Operation, Object}
	Send Pharmacy Message {Operation, Object}

Figure 5.6 High-level statement of permissions and realization of permissions in access control enforcement mechanisms.

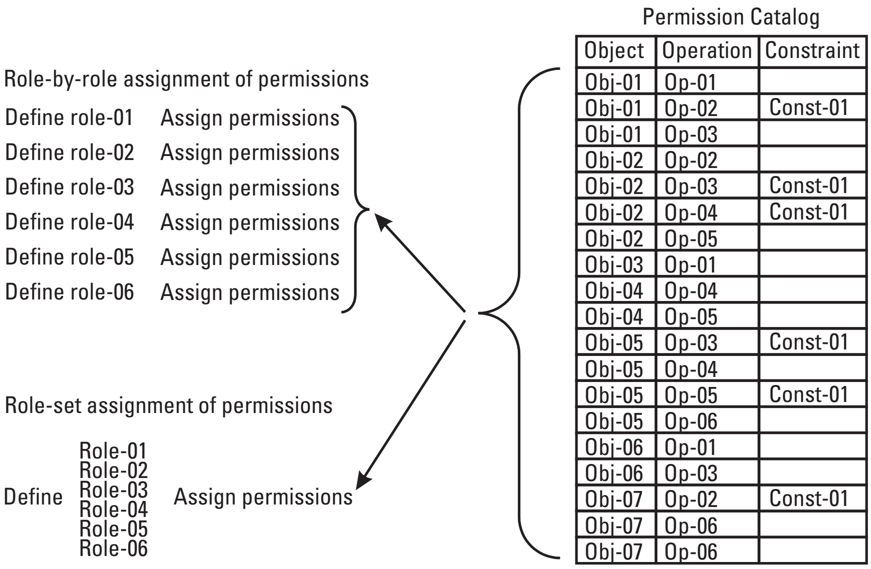


Figure 5.7 Role names defined before assigning permissions versus defining role names one by one.

before assigning permissions to the roles versus defining role names one by one with permissions assigned to them.

Supporting the Access Control Policy

As stated earlier, good roles will support the access control policy. Here we illustrate such a policy to be supported by the defined roles. The access control policy will include objects, potential operations on those objects, and potential roles that can perform those operations on the objects. Table 5.1 provides some examples of access control rules, which as a group comprise an access control policy.

In this table the objects appear in the first column because they are the most fundamental of the components. The operations and role names in the next columns take on meaning in relation to the objects. The operations take on their meaning in relation to the objects to which they are applied. For example, in Submit Invoice "Submit" means "Submit invoice for payment." However, in Submit Order "Submit" means "Submit order for fulfillment."

Table 5.1
An Access Control Policy

Object	Operation	Role Name	Constraint	Note
Invoice	Create	Customer		
Invoice	Display	Customer-service-rep		
Payment	Display	Customer-service-rep		
Catalog	Display	Customer-service-rep		
Order	Display	Customer-service-rep		
Invoice	Display	Customer		
Invoice	Submit	System-admin	ME: Payables-specialist	Submit invoice for payment
Invoice	Save	Customer		
Invoice	Retrieve	Customer		
Order	Create	Customer		
Order	Update	Customer		
Order	Display	Customer		
Order	Submit	System-admin	ME: Payables-specialist	Submit order for fulfillment
Order	Save	Customer		
Order	Retrieve	Customer		
Catalog	Create	Sales-specialist		
Catalog	Display	Sales-specialist		
Catalog	Update	Sales-specialist		
Catalog	Save	Sales-specialist		
Catalog	Display	Customer		
Payment	Display	Customer		
Catalog	Update	Product-specialist		
Catalog	Save	Product-specialist		
Catalog	Retrieve	Customer-service-rep		Permits retrieval from a selection of catalogs
Payment	Receive	Payables-specialist	ME: System-admin	
Payment	Post	Payables-specialist		
Payment	Display	Payables-specialist		
Payment	Reconcile	Account-specialist		
Payment	Update	Account-specialist		
Payment				
Payment				

ME = Mutually exclusive with.

This illustrates that the fundamental component of an access rule is the object, and the operations take on their particular meanings in relation to the objects to which they are applied. In turn, the operation-object (i.e., permission) pairs are more fundamental units than are the role names. This is because the operation-object (permission) pairs can be reused to form parts of various role name-permission pairs. In RBAC terminology, permissions can be assigned to multiple roles. The only constraints illustrated in Table 5.1 are mutually exclusive roles. Permission-permission constraints can also be defined as part of the policy. Not included in the table are user-role constraints (e.g., static separation of duty constraints). These user-role constraints must be defined as well, as part of the access control policy. In practice, the enforcement of these user-role constraints will occur within the administrative tools that are used to assign users to roles, rather that in a runtime system as is the enforcement of other types of constraints. Figure 5.8 shows that most constraints are enforced at run time, but that user-role constraints are enforced statically, in administrative tools.

Business Rules and Security Rules

Access control policy typically includes both business rules and security rules. For several reasons, we recommend that these two be kept separate to the extent possible. Some of the reasons include:

- Security rules are subject to independent review and possible security certification, while business rules are not necessarily subject to the same.

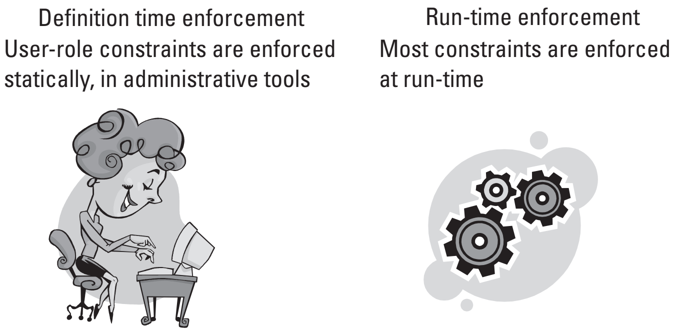

Definition time enforcement
User-role constraints are enforced statically, in administrative tools
Run-time enforcement
Most constraints are enforced at run-time

Figure 5.8 Most constraints are enforced at run time, but user-role constraints are enforced statically. (Clip art copyright clipart.com.)

- Violations of security rules could compromise the viability of the organization, while violations of business rules may not.
- Combining security rules with business rules will make the result more complex and difficult to comprehend in assessing security posture; conversely, keeping security rules separate simplifies their understandability and leads to more trustworthiness.
- Software developers are free to model and implement business rules in the most effective way they can without the need to consult security policy; security rules must be treated with more attention to the security policy.

There will be instances of business rules that are subject to external scrutiny and that can compromise the viability of the organization—for example, compliance with the Sarbanes-Oxley Act, the Health Insurance Portability and Accountability Act (HIPAA), and other government regulations. In such cases it may be necessary to include business rules with security rules. Figure 5.9 shows aspects of security rules and business rules with examples and also shows separation of the two with differing dependencies (e.g., certification and accreditation and freedom of design).

Security rules are subject to independent review and possible security certification	Business rules are not necessarily subject to independent review and security certification
Violations of security rules could compromise the viability of the organization	Violations of business rules may not compromise the viability of the organization
Certification and accreditation requirements are based on security rules	Not being based on security rules provides more freedom of design in the development process
Being based on security rules limits freedom of design in the development process	

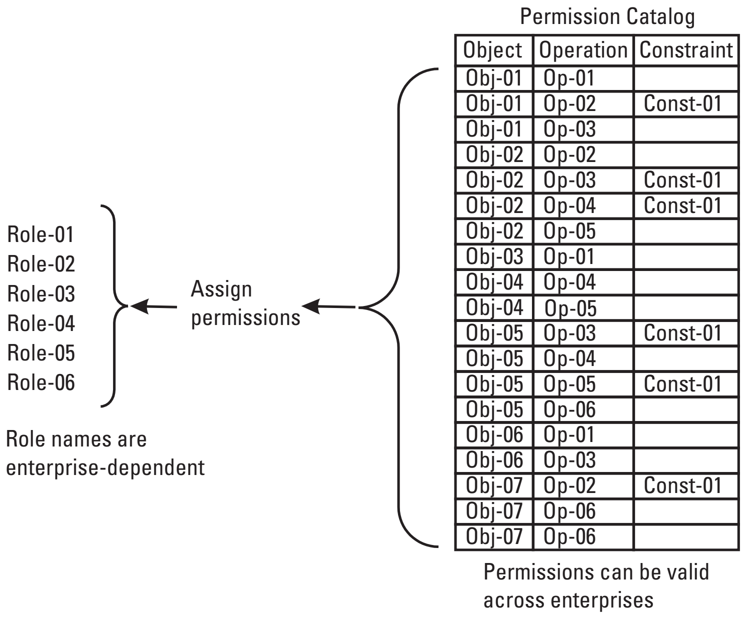
Figure 5.9 Aspects of security rules and business rules; also separation of these with differing dependencies. (Clip art copyright clipart.com.)

Permissions

As stated earlier, permissions are more fundamental than role names, in that a permission can be assigned to multiple roles. Therefore, permissions may be considered as building blocks from which roles may be constructed. The VHA/HL7 work reflects this, their goal being to define a standard set of permissions for the healthcare domain that can be used by organizations in defining their roles. Thus, permissions are considered to be largely the same across organizations, while role definitions are highly organization-dependent. Figure 5.10 shows a permission catalog from which roles are constructed by assigning permissions to role names. Permissions can be valid across enterprises while role names are enterprise-dependent. Permissions should be attached to each role name in turn.

More on Role Names

With regard to selecting role names, several considerations should be noted. As mentioned earlier, role names should be readily recognizable by the


Permission Catalog
Role-01
Role-02
Role-03
Role-04
Role-05
Role-06
Assign permissions
Role names are enterprise-dependent
Permissions can be valid across enterprises

Object	Operation	Constraint
Obj-01	Op-01	
Obj-01	Op-02	Const-01
Obj-01	Op-03	
Obj-02	Op-02	
Obj-02	Op-03	Const-01
Obj-02	Op-04	Const-01
Obj-02	Op-05	
Obj-03	Op-01	
Obj-04	Op-04	
Obj-04	Op-05	
Obj-05	Op-03	Const-01
Obj-05	Op-04	
Obj-05	Op-05	Const-01
Obj-05	Op-06	
Obj-06	Op-01	
Obj-06	Op-03	
Obj-07	Op-02	Const-01
Obj-07	Op-06	
Obj-07	Op-06	

Figure 5.10 A permission catalog is consulted to provide permissions to be attached to role names.

personnel who will be assigning users to roles. Role names first of all should not be abstract identifiers but should model their counterparts in the real world. Further, the names should be meaningful in the particular organization in which they will be used. Thus, the role names should fit naturally into the organization and not be alien identifiers that would have to be carefully researched before the roles can be positively assigned to users.

Where multiple roles exist whose names are similar, the names should be of sufficient length that the different roles can be easily distinguished. That is, brevity of role names should not be a significant criterion, and sufficient information should be included in the name. On the other hand, if too many roles are defined with similar names, perhaps at least some of them should be combined and the differences reflected in another manner, possibly in business rules.

In determining which role names will be readily recognizable by the assigning personnel, it is advisable to elicit candidate names from a selection of these personnel. They should be asked to supply the names of job functions or other role-related identifiers that relate to their particular environments. When a representative sample of appropriate personnel has been canvassed, the resulting candidate names can be assembled and processed to resolve duplicates and produce a workable set of role names. The organizational units where the particular names originated should be maintained with the names for reference.

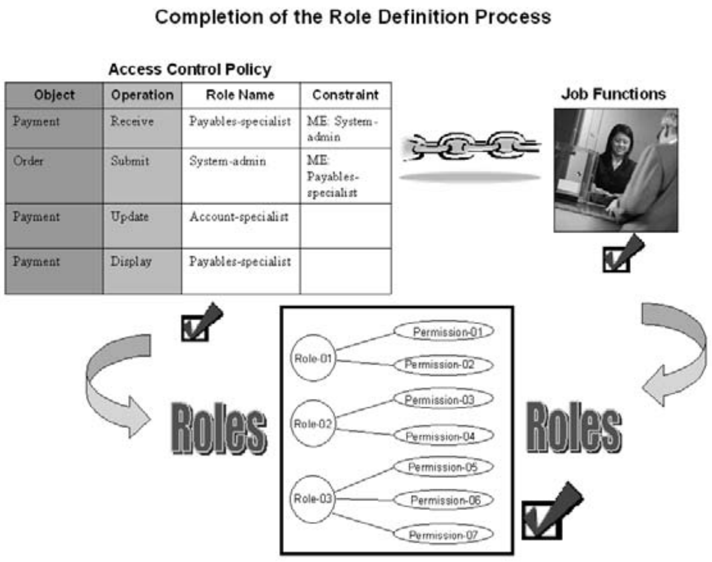
More on Permissions

Since permissions are considered to be building blocks for constructing roles, they should be in the nature of independent units. Selecting one permission for use in a role should not imply selection of specific other permissions. This is different from the existence of constraints among permissions, where selection of a permission may preclude the selection of another specific permission.

As developed by subject matter expert (SME) teams, the permissions will be high-level ones. That is, high-level actions on high-level objects will be defined. For example, a clinical domain permission might be "order laboratory test." The role engineers with system developers will then analyze these high-level permissions into successively more detailed action on objects. Ultimately, these will all be in the IT domain. For example, a detailed permission might be "create laboratory record." Once the detailed permissions have been defined, the high-level name for the permission should be used to cover the entire hierarchy of action-object pairs from high-level to detailed permissions.

When Are We Done?

When defining roles, how do we know when we are done? One indication is whether all of the access control policy has been covered by the role structure. Each element of the policy can be mapped to a role and verified as to its accuracy and completeness. Another indication is whether all job functions identified in the work flow analysis are reflected in the role structure. There will probably not be a one-to-one mapping between job functions and roles because different roles with the same sets of permissions may have been combined. If role hierarchies have been defined, each job function should map to a single role, while that role's effective permissions may be inherited from junior roles. Figure 5.11 shows mappings between access control policy, job functions, and role definitions, with an indication of the completion of the role definition process.



Object	Operation	Role Name	Constraint
Payment	Receive	Payables-specialist	ME: System-admin
Order	Submit	System-admin	ME: Payables-specialist
Payment	Update	Account-specialist	
Payment	Display	Payables-specialist	

Figure 5.11 Mapping between access control policy, job functions, and role definitions. (Clip art copyright clipart.com.)

6

The Role Engineering Process

Roles must reflect the organization's job functions and any other characteristics considered to be relevant for access control. In addition to job functions, other role sources may include responsibilities, organizational position, and authority. It is recommended that a minimal set of roles be defined to begin the role engineering process. This will provide a baseline of experience and results that will serve as a foundation for subsequent efforts.

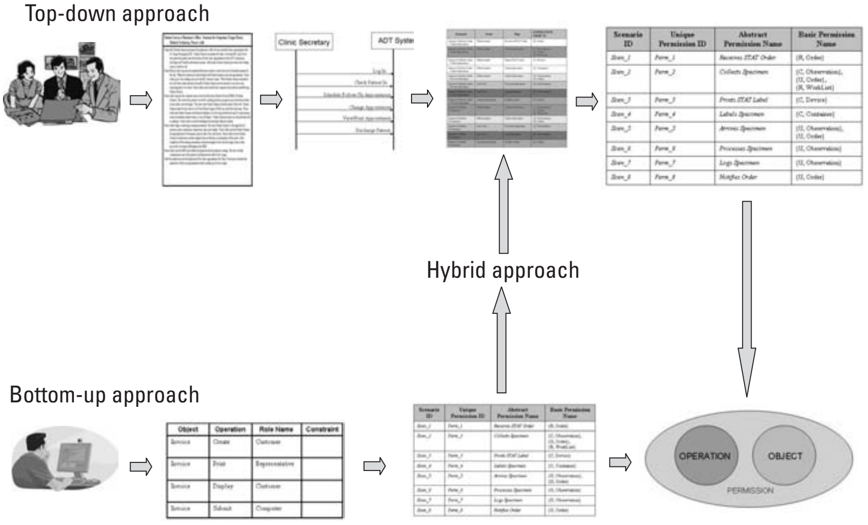
Not following a defined process to identify role names and associated permissions can lead to wasted effort and an inadequate role structure. This in turn can give rise to the idea that RBAC is not a viable option or that the team that defined the roles was somehow inadequate. Prior to these judgments having been made, the inadequate role definitions would have been implemented in IT systems and found to be less than effective. This would likely occur over a period of time and involve a number of people at different levels of the organization. The bad news concerning the roles defined would have spread within and perhaps outside of the organization—not a desirable state of affairs. Of course, these outcomes are to be avoided absolutely. Avoiding them involves using proven approaches and methods to accomplish role engineering.

Approaches to Defining Roles

Given that it is best to follow a defined process in defining roles, how should we proceed? Chapters 12 and 13 address the project management aspects of a role engineering effort. Here we concentrate on how to approach the effort technically and in business terms. Roles may be defined using a top-down approach, a bottom-up approach (also called role mining), or a combination of these two approaches. Figure 6.1 illustrates the top-down approach, the bottom-up approach, and a combination or hybrid approach.

How are roles to be defined? First, we ask, "How do we identify appropriate role names?" The names may come from a human resources department, existing IT roles implicit in application user interfaces, standard lists of role names, and analysis of IT scenarios or use cases. All of these sources except the last one, analysis of scenarios or use cases, constitute a bottom-up approach. Here, existing role definitions, or at least role name definitions, are assumed to have been previously defined and possibly to be valid for adoption across the enterprise. This assumption may be valid in that over time the organization has done some analysis and with practical experience has arrived at the role names and possibly role-permission assignments.

The last mentioned source, scenarios, makes no assumption regarding existing role names or role definitions. In this case we are using a process to discover what the actors are and what the permissions are that these actors need to perform their job functions. This process is similar to one of discovering requirements for a system by using facilitated sessions to identify and



Scenario ID	Unique Permission ID	Abstract Permission Name	Basic Permission Name
Scen_1	Perm_1	Receives STAT Order	(R, Order)
Scen_2	Perm_2	Collects Specimen	(C, Observation), (U, Order), (R, WorkList)
Scen_3	Perm_3	Prints STAT Label	(C, Device)
Scen_4	Perm_4	Labels Specimen	(C, Container)
Scen_5	Perm_5	Arrives Specimen	(U, Observation), (U, Order)
Scen_6	Perm_6	Processes Specimen	(U, Observation)
Scen_7	Perm_7	Logs Specimen	(U, Observation)
Scen_8	Perm_8	Notifies Order	(U, Order)

Object	Operation	Role Name	Constraint
Service	Create	Customer	
Service	Print	Representative	
Service	Display	Customer	
Service	Submit	Computer	

Figure 6.1 Approaches to role definition. (Clip art copyright clipart.com.)

then optimize what is needed to be able to design the system. The process depends on consensus among the members of a team as well as sufficient organization to ensure that results are adequately captured and analyzed. Figure 6.2 illustrates the sources of role names.

Next, we know that permissions must be defined to be able to complete a role definition. A set of permissions must be assigned to each role name. These permissions will reflect the protected object part of the access control policy. When coupled to role names, the resulting roles will reflect the remainder of the access control policy.

Permissions should be defined in relation to roles. This is an important principle. That is, the candidate role names should be taken one at a time and the permissions appropriate to each role name should be defined. Alternatively, all permissions would be defined, all role names would be defined, and then the permissions would be assigned to the roles. What this approach would miss is that the definition of permissions depends on the roles that will be ultimately assigned to the permissions. This does not mean that permissions will actually be assigned to roles during the process of defining permissions, but rather that permissions will be created and engineered in relation to the candidate roles names to which the permissions could be assigned.

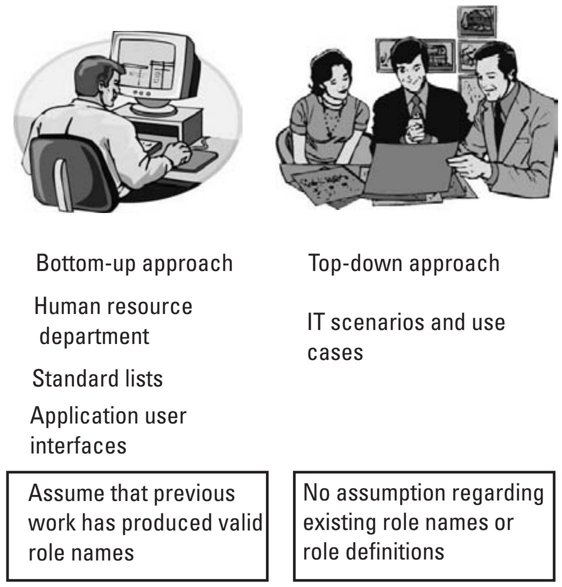

Bottom-up approach
Human resource department
Standard lists
Application user interfaces
Assume that previous work has produced valid role names
Top-down approach
IT scenarios and use cases
No assumption regarding existing role names or role definitions

Figure 6.2 Sources of role names. (Clip art copyright clipart.com.)

When the catalog of permissions has been established, and the role names have been defined, it will be possible to make the permission-role name assignments. Thus, attachment of the permissions to the role names to create roles will be done in a separate phase of the effort even though the role names had been previously used in defining the permissions. Since the permissions will have been defined in light of the candidate role names, the assignment of the permissions to the final role names will not be overly difficult. Only where there are role names that were not considered in the process of defining the permissions will there be a disconnect between the two sets. These cases will have to be elaborated by identifying the outlying role names with other role names and substituting one for the other, or perhaps by defining additional permissions that correspond to these outlying role names.

With the top-down approach, we start from requirements and successively refine the definitions to result in role names, permissions, and roles. Typical steps for the top-down approach are as follows:

1. Create or adopt scenario descriptions of workflows—include actors, actions, and objects.
2. Abstract out the actors, actions, and objects—the actors will be realized as role names and the action-object pairs will be realized as permissions.
3. Work with subject matter experts to define permissions corresponding to workflows.
4. Cluster permissions into sets that correspond to workflows.
5. Name the workflows according to the actors that use the permissions.

While one role engineering process is not expected to be optimal for every role engineering effort, it is useful to present a process for the top-down approach that will work. This process is tied to real-world scenarios that represent typical activities of users of IT systems. In focusing on what users do in the course of their jobs it is possible to deduce what permissions they need within the IT systems or systems. These permissions are not of a technical nature, such as accessing files and running automated processes, but are on the same level at which users think of the permissions, such as adding to a record or computing a score. Once the top-down role engineering process and its components are understood, it will then be possible to make modifications to it as required to accommodate particular requirements.

Our general top-down process is based on one developed by Neumann and Strembeck [1]. The process provides an approach to role engineering that is based on the analysis of IT scenarios, which are depictions of real-world sequences of activities in which various computer users (agents) carry out their job functions in a typical setting. Some of the activities have nothing to do with IT systems and others do involve the use of IT systems. Again, the strategy is to observe the job functions of system users to discover what sets of permissions must exist to support a set of job functions for a given user. In their research paper [1], Neumann and Strembeck do not illustrate a scenario in the form that we have been discussing scenarios. In our view, scenarios are narrative descriptions of the activities of people in a work environment whose activities include some IT use. These descriptions would then be modeled as UML sequence diagrams [2]. Neumann and Strembeck present sequence diagrams initially without showing how they can be derived from narrative scenario descriptions.

The process consists of a series of steps, which are described here. The steps are divided into two parts: scenario modeling and defining the RBAC model. Scenario modeling is an iterative process that is performed until the scenario model is complete. Subsequently, the defining the RBAC model process is carried out.

Scenario modeling includes the following steps:

1. Identify and model new usage scenarios.
2. Derive permissions from scenarios.
3. Identify constraints.
4. Refine scenario model.

Defining the RBAC model includes the following steps:

1. Define tasks and work profiles.
2. Derive preliminary role hierarchy.
3. Define RBAC model.

In our process we concentrate on scenario modeling and the first four steps listed above. Step 3, identify constraints, is performed by recording constraints, among permissions or between permissions and role names, as permissions are defined and as the defined permissions are associated with candidate roles. Step 2 of defining the RBAC model, derive preliminary role hierarchy, brings in the concept of role hierarchies. Use of hierarchies is addressed in Chapter 7. Step 3, define RBAC model, is addressed in Chapter 10.

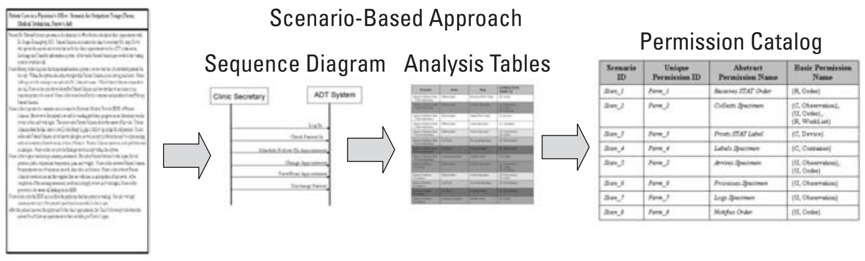
Figure 6.3 illustrates the scenario-based approach to role definition.

Advantages and Disadvantages

There are advantages and disadvantages to this approach. The main advantage to the top-down approach is that a global view of the organization is used, with the likelihood that the resulting roles will be globally valid within the organization. Possibly the primary disadvantage of the top-down approach is that it tends to be time-consuming and requires the time of subject matter experts, which might be difficult to secure for this purpose. A process derived from the Neumann and Strembeck process was developed by the VHA and HL7 [3]. It illustrated the validity of the fundamental Neumann and Strembeck scenario-based process. To keep things simple, the VHA decided not to define hierarchies and constraints initially. It was believed that including these could overly complicate the activities of the available task force members and possibly jeopardize an otherwise successful outcome. Furthermore, in working through the process, some areas for streamlining were noted.

The Scenario Hurdle

While the approach of using scenarios of business operations to derive permissions and constraints is well understood, the fact is that scenarios often do



Scenario ID	Unique Permission ID	Abstract Permission Name	Basic Permission Name
Scen_1	Perm_1	Receives STAT Order	(R, Order)
Scen_2	Perm_2	Collects Specimen	(C, Observation), (U, Order), (R, WorkList)
Scen_3	Perm_3	Prints STAT Label	(C, Device)
Scen_4	Perm_4	Labels Specimen	(C, Container)
Scen_5	Perm_5	Arrives Specimen	(U, Observation), (U, Order)
Scen_6	Perm_6	Processes Specimen	(U, Observation)
Scen_7	Perm_7	Logs Specimen	(U, Observation)
Scen_8	Perm_8	Notifies Order	(U, Order)

Scenario Modeling

1. Identify and model new usage scenarios
2. Derive permissions from scenarios
3. Identify constraints
4. Refine scenario model

Defining the RBAC Model

1. Define tasks and work profiles
2. Derive preliminary role hierarchy
3. Define RBAC model

Figure 6.3 Scenario-based approach to role definition. (*Source:* [1].)

not exist in a given situation. In these cases, suitable scenarios would need to be created for the purpose. To do so requires the availability of subject matter experts (SMEs) and a facilitator. The first step is to identify a typical scenario setting and purpose, such as a clinical encounter with a patient or loan application by a bank customer. The next step is to state the activities one by one that would take place within the scenario. Some scenarios writers use fictitious names for the people in the scenarios to make them more descriptive. This is not a necessity, so long as the various actors in the scenario are identified in some way, for example by their job title.

In practice, a group of SMEs working with the creation of scenarios often comes to a realization that scenarios contain substantive information and also more superfluous information that does not contribute to the usefulness of the scenario. In addition to names of persons, the superfluous information may be such things as movement from one place to another, using a particular device, entering or leaving a room, and waiting for something to be available. Thus, a desire can arise to capture the essentials of a scenario, for future processing into permissions and constraints, while eliminating the superfluous (and variable) aspects of the scenario. This can, in fact, be done, but it requires a team of SMEs who have become familiar with the generation of scenarios and then using the scenarios to derive permissions and constraints. Essentially, the group will mentally create scenarios and discuss these among themselves and record the essential components of the virtual scenario. This short-circuiting of the scenario-based approach is illustrated in Figure 8.5.

To illustrate, a scenario is presented here with a corresponding distillation of the scenario. It is the distillation that would be mentally envisioned and recorded by the team. The example is taken from the VHA/HL7 work [4].

A distillation of the scenario might take the form illustrated in Table 6.1.

This distillation illustrates the conclusion often reached by a role engineering team that the development of a full scenario, while useful as a point of departure, entails a considerable amount of work and time, with the essential role engineering content forming a small fraction of the content of the scenario. Thus, the desire to eliminate the development of scenarios and to proceed as though a scenario had been written becomes apparent. In light of this desire to short-circuit the scenario-based process, we must answer the question as to what might be lost by short-circuiting the process by eliminating the narrative scenario for the process.

Microbiology Laboratory Observation Event (POLB_SN000900)

Preliminary Activities

Dr. Eric Emergency, an emergency room physician, sees a 45-year-old male diabetic patient, Adam Everyman, for an abscess on the right foot with necrotic tissue.

Dr. Emergency collects a tissue sample of the wound discharge and orders a Gram Stain and Wound Culture with Sensitivity. The order is sent from the Order Management System (OMS) to the Laboratory Information System (LIS). The sample is transported anaerobically to the laboratory where the order is waiting.

A Medical Laboratory Technician (MLT) Fred Collector, who works in the specimen-processing section of the laboratory, accessions the specimen into the Laboratory Information System LIS. The LIS notifies the OMS that the specimen has been received and that it intends to perform the requested series of tests and transmit the specimen accession number. Fred labels the specimen with the specimen label which is printed on the Specimen Label printer located in the Specimen Processing Section. The Media Labels automatically print out in the Microbiology Section. Fred transports the labeled specimen to the Microbiology Section, where MLT Harry Planter inoculates the specimens onto Blood Agar Plate (BAP), Chocolate Agar Plate (CHOC), MacConkey Agar Plate (MAC), and Thioglycollate Broth (Thio). Harry then makes a smear on a glass slide and performs a Gram Stain.

Initial Results

The Gram Stain is read by Nancy Sellia and the result of the preliminary testing is transmitted from the Microbiology System Module to the Results Management System (RMS). The Gram Stain report includes the following elements:

- 3+ PMN's (polymorphonnuclear granulocytes);
- 3+ Gram Positive Cocci;
- 2+ Gram Negative Bacilli;
- 1+ Gram Positive Bacilli, with terminal spores.

Note that the initial results can be transmitted using a standard laboratory event message as well as the microbiology report.

The next day, Nancy Sellia reads the inoculated media. On the Blood Agar Plate (BAP) she has small pinpoint colonies surrounded by a zone of beta hemolysis. An agglutination test confirms that it is a Group A Streptococcus and sensitivities are inoculated with this isolate and incubated.

On the MAC she observes smooth round colony that is pink in color. An Enteric panel and sensitivity is set up and incubated on this colony.

The Thio tube shows moderate growth in the middle part of the tube. The Thio is subcultured to an anaBAP, kanamycin/vancomycin laked blood (KVLB), anaPEA agar plates. A Gram Stain is performed and Nancy sees Gram Positive Bacilli with spores under the microscope.

Preliminary Results

A preliminary culture report is released by Nancy, and the result (POLB_IN004110) is transmitted from the Microbiology System Module to the Results Management System (RMS) to the Order Management System. Preliminary culture report is as follows:

- 3+ Lactose Fermenting Gram Negative Bacilli;
- 2+ Group A Streptococcus;
- 2+ Gram Positive Rods with spores noted in Thioglycollate Tube;
- Identification and Sensitivity to follow.

As indicated above, the preliminary results can be transmitted using a standard laboratory event message as well as the microbiology report.

The next day, Nancy Sellia reports that Group A Beta Streptococcus is sensitive to Pencillin.

Certain antibiotics are retained in the microbiology module and not reported to the LIS or OMS.

The LF colony is identified as Escherchia coli: resistant to Ampicillin; susceptible to Cefazolin, Ciprofloxacin, Nalidixic Acid, Sulfisoxazole, conjugative transposable element (SXT), and Nitrofurantoin. The panel biochemical reactions and certain antibiotics are retained in the microbiology module and not reported to the LIS or OMS.

On the anaBAP a small pinpoint colony with a double zone of beta hemolysis was isolated. An anaGram Positive panel and sensitivity was set up and incubated anaerobically.

A preliminary culture report is released by Nancy, and the result is transmitted from the Microbiology System Module to the Results Management System (RMS) to the Order Management System. The preliminary culture report is as follows:

Preliminary Microbiology Results

- +3 Escherichia coli
  - Resistant Ampicillin
  - Resistant SXT
  - Susceptible Cefazolin
  - Susceptible Ciprofloxacin
  - Susceptible Gentamicin
  - Susceptible Nitrofurantoin
- +2 Group A Beta Hemolytic Streptococcus (Streptococcus pyrogenes)
  - Susceptible Penicillin
- +2 Anaerobeic Gram Positive Bacilli with double zone of hemolysis
- Identification and Sensitivity to follow

Intent to Perform Occurrence

The two days later, Nancy Sellia reads the anaGram Positive panel and sensitivity. She identifies the colony as Clostridium perfringens sensitive to Penicillin. The panel biochemical reactions and certain antibiotics are retained in the microbiology module and not reported to the LIS or OMS.

Final Results

- +3 Escherichia coli
  - Resistant Ampicillin
  - Resistant SXT
  - Susceptible Cefazolin
  - Susceptible Ciprofloxacin
  - Susceptible Gentamicin
  - Susceptible Nitrofurantoin
- +2 Group A Beta Hemolytic Streptococcus (Streptococcus pyrogenes)
  - Susceptible Penicillin
- +2 Clostridium perfringens
  - Resistant Pencillin
  - Susceptible Clindamycin
  - Susceptible Metronidazole
  - Susceptible Piperacillin

Table 6.1
A Scenario Distillation

Title: Microbiology Laboratory Observation Event (POLB_SN000900)
Location: Emergency Room, Laboratory
IT Accesses by users: Order Manageent System (OMS), Laboratory Information System (LIS)—Microbiology System Module, Results Management System (RMS)
Access Modes: NOT STATED

One attribute of the narrative scenario is that it reflects a real-world operational setting where a user conducts job activities to accomplish a task within the scenario. So if the narrative were to be eliminated, this real-world view would be lost. In reality, if the role engineering team eliminates the documentation of scenarios, it still creates a mental image of the scenario and uses the mental image to identify permissions that correspond to candidate role names. So the question becomes how important is it to have a documented scenario as part of the process.

Our opinion is that it is not as important to have a scenario written down so much as to have used a mental image of a scenario to elicit data on actors and corresponding operations on objects. For example, we note that N&S do not refer to narrative scenarios in their paper. This does not, of course, imply that they do not believe a narrative scenario should be used to define a sequence diagram, only that the narrative scenario is not an integral part of the process. As noted, the team can still proceed to distill out the essential components needed to identify permissions and constraints. In our view, as stated above, the concept of the scenario is needed by the team to be able to reach a valid distilled result. If the team decides not to write down complete scenarios, a record should be maintained of each mental scenario used, along with the distilled results; for example, a scenario title, a one- or two-sentence summary of the scenario, and notes on any discussion of the scenario carried out among the team members prior to their arriving at the final results.

Regarding the scenario above, we note a lack of detail regarding the nature of the accesses to the IT resources. In this scenario, only the names of the information systems are provided. This illustrates a potential pitfall in developing scenarios: even after expending the time and effort to create a scenario, it may be deficient in information content. This lends further support to discussing an unwritten scenario among the team and then distilling the essential IT access features. A standard template of categories similar to the one in Table 6.1 can be defined to ensure that all essential information is recorded. The work is greatly reduced and the results can be more complete than might be the case after writing a scenario.

With the bottom-up approach, we examine existing information systems and extract implicit role definitions from such items as screen contents, user profiles, and access control lists. Typical steps for the bottom-up approach are as follows:

1. Examine existing systems that perform access control.
2. Identify types of users (e.g., according to profiles available to be assigned to users).
3. Identify the permissions that the users can perform using their assigned profile(s).
4. Use this information to represent prototype roles.

A degree of creativity is needed to abstract role name and permissions from the available artifacts. It is by no means a process that can be followed without analytical skills to recognize role names and permissions. Once roles have been identified using a bottom-up approach, it is still necessary to normalize the results and transform the results into meaningful roles for the organization. The main advantage of the bottom-up approach is that it capitalizes on existing work products and potentially avoids some current work to develop the same results. Possibly the primary disadvantage of the bottom-up approach is that it still requires a degree of work to accomplish and it is not likely that the results will reflect a valid global view of the organization. Also, it will be necessary to reconcile the role data obtained from the bottom-up approach with other role data, including that obtained from a top-down approach. This process in itself can be time-consuming and its results may not merit the work required.

In practice it may be advantageous to combine the two approaches, top-down and bottom-up. The decision to use one or the other depends on several factors. Ideally, the top-down approach would be used exclusively, since it results in the most globally defined roles. In cases where the roles in existing systems are considered to be very reflective of the goal set of roles to be implemented, it could be advisable to use a bottom-up approach exclusively.

A Recommendation

The approach recommended here is to aim for a top-down approach and to assess the available material in existing systems to determine its potential in providing quality role information. If this does not appear to be the case, we recommend not spending the resources for any bottom-up work, as the likelihood of producing useful results would not be very high where existing systems will not provide quality role definition material. In any case, it is not likely that the bottom-up results will reflect a valid global view of the organization. Also, it should be remembered that any bottom-up results will still need to be reconciled with the top-down results, resulting in additional work needing to be done to combine the two approaches. This additional work may be better spent on the top-down approach.

References

[1] Neumann, G., and M. Strembeck, "A Scenario-Driven Role Engineering Process for Functional RBAC Roles," *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, Monterey, CA, June 3–4, 2002, pp. 33–42.

[2] Alhir, S., *UML in a Nutshell*, Sebastopol, CA: O'Reilly & Associates, Inc., 1998.

[3] Science Applications International Corporation (SAIC), *Role-Based Access Control (RBAC) Role Engineering Process Version 3.02*, San Diego, CA, January 20, 2005.

[4] VHA/HL7 Collaboration, *Microbiology Laboratory Observation Event* (POLB_SN000900), June 10, 2003, http://www.va.gov/rbac/documents.asp.

7

Designing the Roles

Chapter 5 discussed defining good roles. The guidelines presented there apply to the design of roles. As stated, "Good characteristics are those that make the roles easier to manage and also support the access control policy effectively." Roles must be designed to simultaneously implement the access control policy and to model the organization's job functions. These two considerations are not necessarily related, and in fact they can be mutually antagonistic.

There are two sources of data for designing roles. For implementing access control policy the source is internal requirements for protecting information objects within IT systems and the allocation of access to the objects, in various forms, to individuals and groups within the organization. For modeling the organization's job functions the source is less defined. If an effort has not already been carried out to define a structure of job functions, these results can be used in establishing a role structure. In the event that the organization's job functions have not already been modeled, an effort to model them will need to be conducted. The sources available to provide the information for this modeling include human resources data, access control data in existing or planned systems (role mining), job function definitions that can be adopted from similar organizations, and job function definitions that can be adopted from existing standards.

Some of these sources of data for modeling job functions may serve to feed the need for role names and definitions of functions performed. These are job function definitions from human resources data, from other organizations, or from existing standards. However, these sources do not provide the tying of role names and definitions to the IT functions performed or with the corresponding access control policy. Of the sources identified, only data from existing or planned systems will provide this tying of role names and definitions to an access control policy.

Given these potential sources of data for defining roles (names, operations, objects), a role engineering effort must be conducted to establish a role structure that simultaneously implements the access control policy and models the organization's job functions. If it is found that existing sources of role definition data are not adequate to meet the needs of the role engineering effort, it would be necessary do conduct a new data gathering effort. This effort may include conducting interviews of users, managers, technical staff, and policy specialists. Another approach to data gathering and analysis is to use facilitated interactions of subject matter experts to identify permissions and possibly role names. This is an ideal approach, although it is labor intensive and that labor is taken from the valuable time of subject matter experts.

Here we note a possible deficiency with the bottom-up or role mining approach to role identification. The results of a bottom-up process will probably not result in roles designed to implement the access control policy and to model the job functions. One reason for this is that for the access control policy and job model to be simultaneously reflected it is necessary to define the permissions in light of the candidate role names. For example, role mining may indicate that a class of users tends to have a clustered set of permissions. However, each of these clusters, which would later be realized as roles, may not, as a set, reflect the access control policy. If they do not, the role engineers must resolve discrepancies with the policy and perhaps reassign permissions to the clusters representing candidate roles. These will be isolated to a particular system and may be disjointed from a set of permissions mined from a different system.

As defined by the RBAC standard [1], a role consists of a role name ("role" in the model) and the permissions that are assigned to the role name. Therefore, at a minimum, to define a role means to define a role name and its associated permissions. Beyond this, role engineering also includes defining hierarchical relationships among roles and defining constraints. Figure 7.1 illustrates this definition of role engineering.

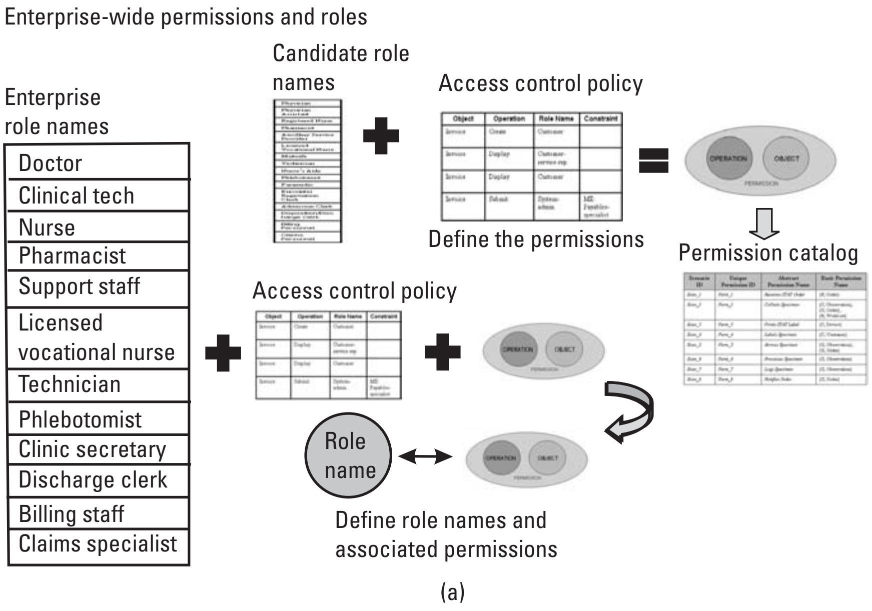
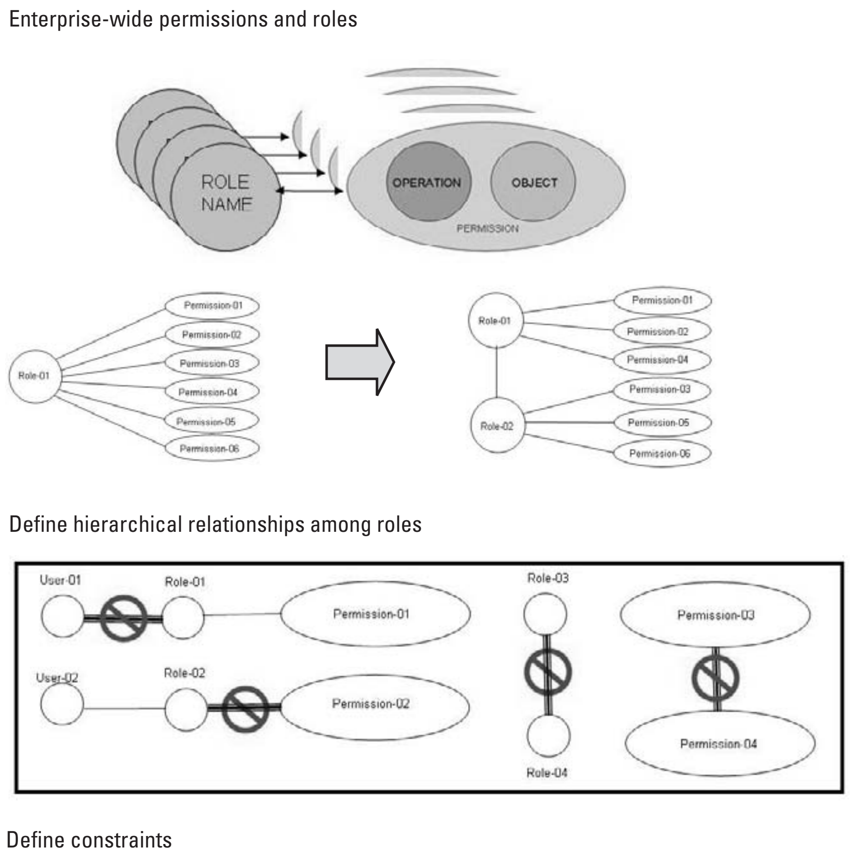


Figure 7.1 (a, b) What is role engineering?

How Do We Go About Engineering Roles?

First we will cover defining criteria for the selection of a role name. Keeping in mind that the role names will ultimately be used by administrative personnel to assign users to roles, the role names should be easily recognized and understood within the enterprise by the individuals charged with assigning them to users. Thus, role names must reflect the organization and its personnel activities within IT systems in an intuitive way.

In selecting names for roles, several sources are available. In a few instances, standard lists of role names are available. In healthcare, the ASTM list of "Healthcare Personnel that Warrant Differing Levels of Access Control" is a possible starter set of role names [2]. Other possible sources for starter role names are the Military Occupational Specialties lists and similar ones for the U.S. military, and the Office of Personnel Management Job Classifications for U.S. civilian occupations. In general, it is desirable to use such an established set of names rather than creating a new set, for purposes of semantic interoperability among enterprises. These lists of potential role names in many cases may not reflect the particular job functions for a given organization. In these cases it is not advisable to try to use a name from a



(b)

Figure 7.1 (continued)

standard list if it does not represent an accurate fit. Where the available sets of standard role names are inadequate or suboptimal, one remedy is to work with the appropriate standards and other organizations to improve the set of names. This was done by the Veterans Health Administration with respect to the ASTM list of healthcare role names. When it is necessary to create role names from scratch, through a requirements gathering process, it may be beneficial to attempt to standardize the new role names through an external body.

Using the scenario-based role engineering process, the role names are derived from task and work profiles [3]. At a high level, the candidate role names are based on clusters of permissions that are used by a single user to carry out a particular work flow.

Another desirable property of role names is that they be stable. The obvious reason for this is that permission assignment to roles should not need to be changed often. If the role names are changing at all, a significant amount of rework will be needed to assign permissions to the new role set. Each time a role definition changes—by changing role names, permission assignments, or both—a degree of confusion will inevitably result. This confusion will, in turn, interfere with one of the benefits of RBAC, that of facilitating the audit of user permissions.

A Strategy for Preserving Role Understandability

One strategy that may be employed to mitigate the deterioration of understanding resulting from changing role definitions (name, permissions, and so on) is to keep a role with the previously understood name as the senior role in a hierarchy. Then, it is possible to change some or all the roles junior to that senior role while maintaining the original role name. In some cases it may be desirable to create a new role with an intuitive name to dominate or "cover" a hierarchy of changed roles. Of course, this use of the hierarchy mechanism may introduce a degree of complexity in the role structure that was not originally present. Figure 7.2 illustrates the use of an intuitive role name to cover a hierarchy of changed roles.

It can be expected that some role names within an organization will be more stable than others. For example, in a clinical setting there will always be roles called "Doctor," "Nurse," and "Pharmacist." These stable role names can be used as senior roles in hierarchical relationships of functional roles to promote the understandability of the role names in assigning users to the roles. Additionally, the stable role names can be used for structural roles. These structural roles differ from functional roles in that their permissions are those needed to connect to an IT resource such as a system, work flow, server, or application. The functional roles have permissions that permit functional activities within an application.

Structural Role Names Should Mirror Functional Role Names

Where possible, structural role names should be mirrored among the functional role names. For example, a structural role named "Doctor" should be

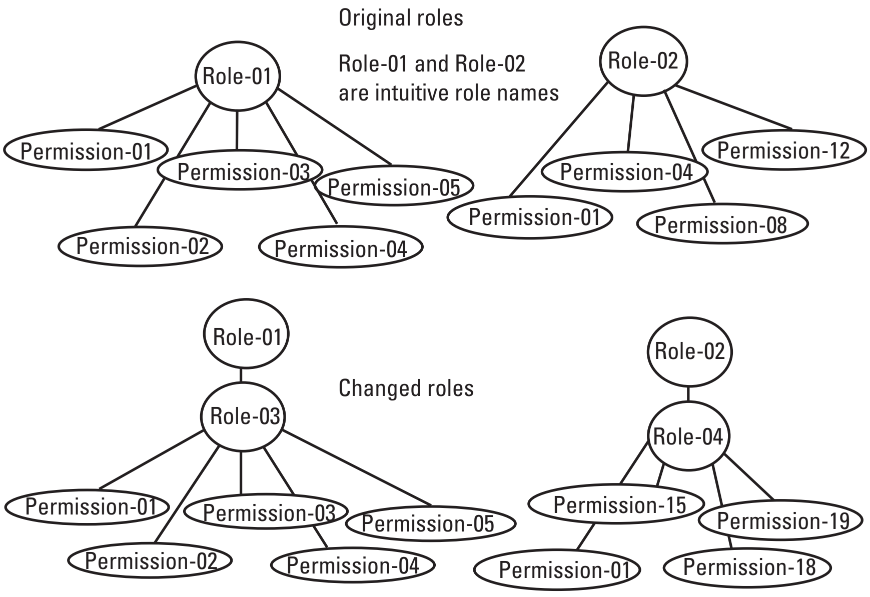


Figure 7.2 Using an intuitive role name to cover a hierarchy of changed roles.

accompanied in the organization's role set by a functional role "Doctor." Ideally, it would only be necessary for the individual assigning users to roles to make one assignment to the role name "Doctor." The IT system would then automatically assign the user to the structural and functional roles "Doctor." The revocation of the user from the "Doctor" roles could subsequently be made to either of the "Doctor" roles. For purposes of revoking a user's access in a central place, it would only be necessary to revoke the structural "Doctor" role to simultaneously revoke the user's functional "Doctor" role.

When to Use Hierarchies

Hierarchical relationships among roles can be defined when it is determined that they will be of value. Hierarchies will be of value where roles exist—before the introduction of a hierarchy—which are made up of similar sets of permissions. Rather that allowing these similar roles to exist separately, it can be more efficient to assign the common sets of permissions to a junior role or roles and then to define senior roles that inherit the common permission sets and then have the uncommon permissions assigned to them. Figure 7.3 illustrates the transformation of a set of roles having similar permission sets to a

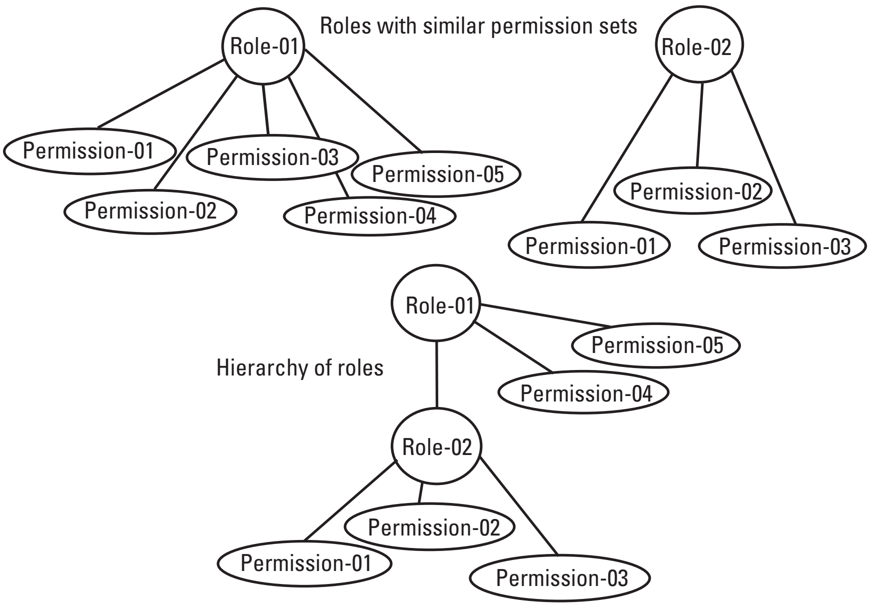


Figure 7.3 Transformation of a set of roles having similar permission sets to a hierarchy of roles.

hierarchy of roles that distribute the permissions among the roles in a more efficient manner. They are a convenience more than a necessity in defining roles for an organization. Hierarchies among roles imply the existence of inheritance properties among roles in a hierarchical relationship. The concept of senior roles and junior roles comes into play. A senior role is higher in a hierarchy and a junior role is lower in that hierarchy.

Inheritance in role hierarchies works as follows. When senior roles are represented higher in a hierarchy diagram, the usual case, role membership is inherited down the hierarchy tree and permissions are inherited up. That is, senior roles inherit the permissions of all their junior roles, whether the junior role is immediately adjacent to the senior role or whether the junior role is below another junior role in the hierarchy. Effectively, junior roles inherit the users assigned to all senior roles, in that the users assigned to the senior roles may equivalently have been assigned to the junior roles with the same effect. Figure 7.4 illustrates the inheritance of permissions by senior roles and inheritance of user assignments by junior roles.

The use of hierarchical relationships also becomes a consideration when certain roles in the organization are common to many users and other roles

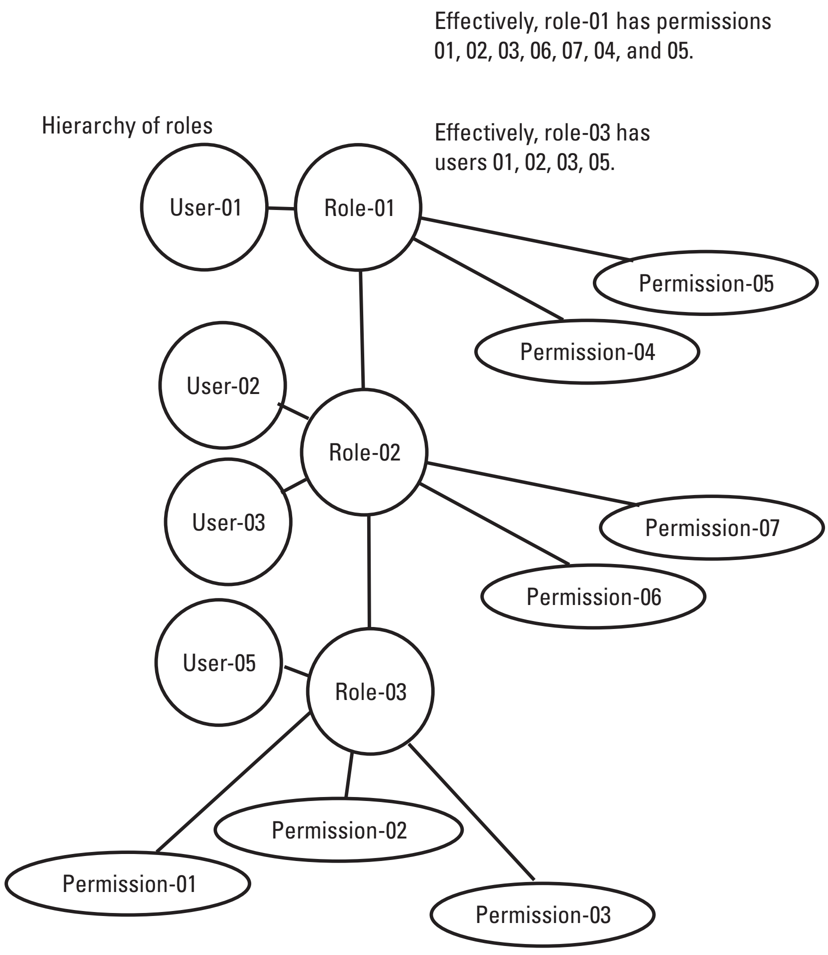


Figure 7.4 Inheritance of permissions and user assignments in role hierarchies.

are common to fewer users. With inheritance, it will be unnecessary to assign users explicitly to each and every role a given user needs. With the commonly held roles lower in a hierarchy and the less commonly held roles higher in the hierarchy, it is possible to assign a user to one of the senior roles and have the related junior roles (actually the permissions assigned to those junior roles) automatically assigned to that user. This is the potential convenience of role hierarchies. Figure 7.5 illustrates this efficiency in user assignment gained by defining common roles as junior roles and defining the less common roles as senior roles.

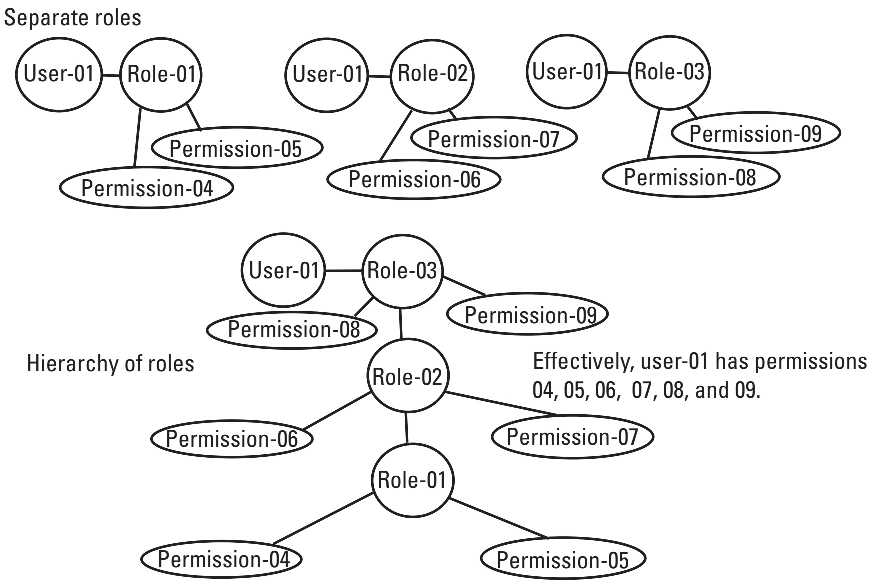


Figure 7.5 Creating efficiencies in user assignment of roles by using hierarchies.

Defining Role Hierarchies

One way to define a role hierarchy is to take a role that contains a diverse collection of permissions and decompose that role into one or more junior roles. This implies that the role engineering staff will make a pass through the current set of defined roles and look for roles that may have been defined with a large number of permissions or with a diverse set of permissions. Of course, "large number of permissions" is considered in relation to the number of permissions as seen from one role definition to another. Concurrent with a relatively large number of permissions on a given role is a potential for violating the security policy's least privilege requirement. That is, the underlying reason for decomposing a role with a large number of permissions into junior roles with fewer permissions is the desire to come closer to the realization of the principle of least privilege.

Another way to define a role hierarchy is to attempt to define the roles to be simple and atomic to the extent possible—that is, never to create a role with a relatively large number of permissions in the first place. Then this pool of roles defined with a small number of permissions can be used to compose more complex roles by incorporating the simpler roles into hierarchical relationships. Tempering this use of simple roles to create hierarchies of roles is the possibility of creating too many roles to the extent that the roles are not intuitively understood by the administrative personnel and others who will be assigning users to the roles.

Role hierarchies should be considered for use when many users will need a similar set of permissions. These permissions can be defined for what will become junior roles and then these junior roles can be placed into role hierarchies where needed. This avoids assigning similar sets of permissions to multiple roles. This is somewhat different from the simplification of user assignment to roles by using hierarchies described above, in that here we are indicating how role definitions can be simplified using hierarchies. Figure 7.6 illustrates this simplification of role definitions by using hierarchies.

Role hierarchies should not be used when a relatively few roles will suffice to support the access control policy. In this situation, using a role hierarchy would add complexity with little benefit in role normalization. A rule of

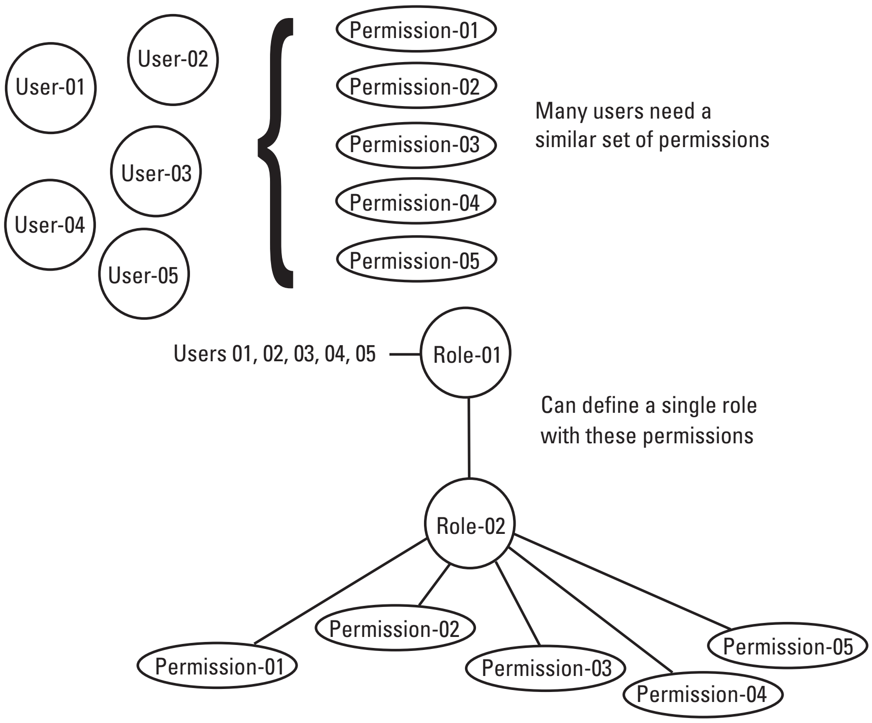


Figure 7.6 Simplification of role definitions using hierarchies.

thumb would be that creation of a hierarchy should redistribute a certain percentage of permissions from a senior role to one or more junior roles. This percentage should be at least 40%. If a smaller percentage of senior role permissions would be distributed to junior roles, not enough of the overburdened senior role's permissions would be redistributed to warrant the complexity added by introducing the hierarchy. Table 7.1 displays values for this 40% distribution of permissions as redistributed to *n* junior roles created from one senior role. The third column in the table illustrates that as the number of new junior roles reaches eight and beyond, the mean percentage of distributed senior role permissions per junior role levels off. This indicates that eight may be the maximum number of junior roles that should be defined from a single senior role.

Potential issues when using hierarchies can arise where large numbers of roles are incorporated into a hierarchy and these role definitions are less than intuitive. For example, it may seem advantageous to define many atomic roles and compose more complex roles from these. This is an efficient way to define a role structure. However, if the roles are too atomic they may lack meaning in the real world and be seldom or never used for user assignment. This is not conducive to administrative efficiency, at least so far as user assignment is concerned. Another potential issue with hierarchies is that the relationships among the roles and their inheritance relationships can become

Table 7.1
Percentage of Senior Role Permissions Distributed to Junior Roles

Number of Junior Roles Created Using Permissions from a Single Senior Role	**Mean Percentage of Permissions per Junior Role**	**Mean Percentage of Distributed Senior Role Permissions per Junior Role**
1	40	100
2	20	50
3	13	32
4	10	25
5	8	20
6	7	18
7	6	15
8	5	12
9	4	10
10	4	10

quite complex and hard to understand. This works against the security goal of simplicity.

Before designing a role hierarchy, it is necessary to define the roles that will participate in the hierarchy. There should be a balance between the number of roles within a given hierarchy and the complexity introduced by the relationships among the roles. Without any role hierarchies it is possible that the number of permissions assigned to each role is excessive, and it is necessary to assign the same bundles of permissions to multiple roles. If this is the case, it should become apparent that a simplification could be achieved by extracting those permission bundles and naming them as separate roles. With too many hierarchical relationships, it is possible that the roles do not represent meaningful roles in the organization but are mere convenient permission bundles. Also, if there are too many hierarchical relationships among roles, it will be difficult for an administrative person to be certain when assigning users to roles that the individual is assigned to the correct combination of roles to achieve the desired permission set for the individual.

Alternatives to Hierarchies

Alternatives to hierarchies exist and should be employed when the complexity of role hierarchies reaches an unacceptable level. One alternative is to define a role for each and every unique set of permissions needed to perform each job function by an assigned user. This can reduce or eliminate the complex hierarchy. At the same time, it can multiply the number of roles defined for the organization, as the hierarchy serves to parse out sets of permissions and establishes a normalized role set. In turn, this multiplication can be mitigated by partitioning the roles into sets that can be administrated by different personnel. A degenerate case would be to define a separate role for each user, which would eliminate hierarchies but also eliminate the administrative advantage of RBAC.

Constraints

Constraints can be relationships between roles (role-role constraints), relationships between permissions (permission-permission constraints), relationships between permissions and role names (permission-role constraints), and relationships between users and roles (user-role constraints).

Various types of constraints can be defined on roles and permissions. General types of constraints include time of day, user location, and number of users assigned to the role (cardinality). Inter-role constraints can include static and dynamic separation-of-duty constraints and prerequisite roles. Interpermission constraints can include mutually exclusive operations and mutually exclusive objects.

In the early stages of role engineering, the need for constraints may not be apparent. Definition of constraints can be postponed to a later stage with no significant loss of efficiency. The need for constraints can be identified through a subsequent requirements gathering and engineering process that assesses the efficacy and efficiency of the role definitions in supporting the access control policy.

Another consideration is that some commercial products may not provide an easy way to implement constraints in RBAC systems in the way specified by a particular access control policy. Therefore, it is advisable to determine the type and degree of support for constraints by available off-the-shelf products before planning any significant effort to identify and design role constraints.

References

[1] American National Standards Institute, *ANSI INCITS 359-2004 American National Standard for Information Technology—Role Based Access Control*, February 3, 2004, http://www.incits.org.

[2] American Society for Testing and Materials (ASTM), *E 1986–98 Standard Guide for Information Access Privileges to Health Information*, November 1998, http://www.astm.org.

[3] Neumann, G., and M. Strembeck, "A Scenario-Driven Role Engineering Process for Functional RBAC Roles," *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, Monterey, CA, June 3–4, 2002, pp. 33–42.

8

Engineering the Permissions

In this chapter we address the permissions that are to be assigned to role names in order to create roles. For the definition of permission we use the one found in the RBAC standard: a permission is an operation on an object. Permissions are the fundamental building blocks for defining roles. They contain the names of protected objects which participate directly in the security policy. If there were no protected objects, there would be no point or need to define roles. Closely associated with the protected objects are the actions permitted on those objects. The actions define the specific modes of access to the associated objects. Therefore, each combination of an action on an object has been defined as a permission. Permissions directly reflect the access control policy.

It is important to get the definitions of permissions correct because permissions are the basic building blocks from which roles are constructed. Permissions must be engineered because they are actually rather complex entities. Permissions must not be defined in a haphazard manner. While they seem to be relatively simple things—merely operations on objects—they are really rather complex in nature. This is because they are composed of three things: operations, objects, and the relations between operations and objects. Therefore, to identify the objects in permissions and then to identify and associate operations on these objects, requires significant effort to create a candidate permission and then to verify the validity of the result with respect to the access control policy.

Objects

In the process of engineering permissions we bear in mind the fact that objects are independent of role names (or roles, in terms of the model in the RBAC standard) in that they form part of permissions. While role names have no semantics in relation to an access control policy if they have no permissions assigned to them, permissions do by themselves have semantics in relation to an access control policy. As stated earlier, however, it makes little sense to define permissions in the absence of meaningful role names. Objects are more fundamental than operations because objects are the reason for having an access control policy at all. The operations on the objects constitute windows upon the objects that allow actors to access the objects in defined ways. Figure 8.1 illustrates the relationships among objects, operations, and role names.

The objects appearing in permissions definitions will be assigned names that are global within the high-level permission definitions. These names will not be the same as the names that appear in low-level permissions within IT systems. The properties of the objects in high-level permission definitions are not of great importance, the main requirement being that the operations on each object have semantic reality in conducting the business of the enterprise.

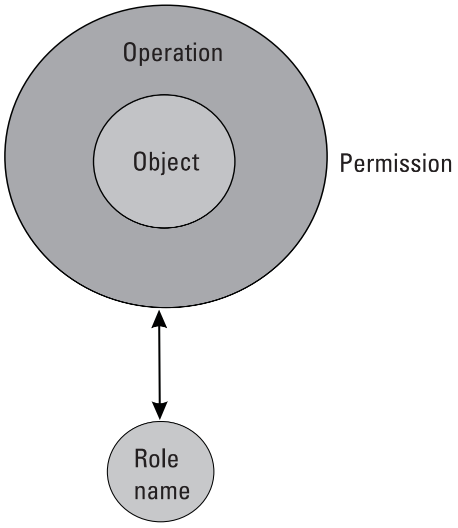


Figure 8.1 Relationships among objects, operations, and role names.

Operations

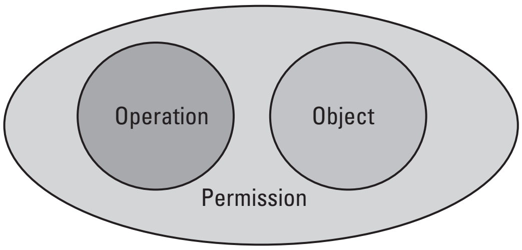
Within a scenario such as the example that appears later in this chapter, there will be both real-world actions and IT system actions. Only the IT actions will be used to define permissions. Because of the high-level definition of permissions in the permission engineering process, the IT operations entailed by a high-level permission can be multiple in nature. The translation of the high-level permissions to IT permissions will be conducted by role engineering staff and system developers.

Operations on Objects

When we associate operations with objects to form permissions, the sum of the parts is greater than the individual components. That is, the fact that an operation is associated with an object is a piece of information in itself. Not all operations will be valid on all objects. The process of associating operations with objects will generate information on which operation-object combinations are permitted or denied.

Levels of Abstraction

With the definition of a permission as an operation on an object, a permission can be defined at any level of abstraction that has meaning to the organization in the context of an IT system or network. That is, a permission can have meaning to a nontechnical user of a system, to a manager, to an administrative person, or to a system designer. Since a permission for an IT system must ultimately be implemented in an actual system, the higher level or abstract permissions must be capable of being made more detailed and implementable in an IT system or systems. For example, a high-level permission might be "Enroll a new employee in an insurance plan." The operation would be "Enroll" and the object would be "Employee." The other parts of the high-level permission, pertaining to "new" employee and "in an insurance plan" would be additional attributes and context that would be implemented in the IT system, but not be part of the permission itself. A low-level permission might be "Read vital signs record," where the operation would be "Read" and the object would be "Record." The "vital signs" portion would be an additional attribute that would be implemented in the IT system. Figure 8.2 illustrates permissions defined at several levels of abstraction.



Level	Operation	Object
Business	Submit	Invoice
IT Requirement	Store	Invoice Contents
IT Design	Update	Invoice Table(s)
IT Implementation	Open Read Insert Read Verify Commit	Invoice Row

Figure 8.2 Permissions defined at several levels of abstraction.

Permissions Are Independent Building Blocks

Is it possible to define permissions separate from the ultimate roles to which they will be assigned and also have permissions that have meaning within the organization? As stated earlier, permissions are building blocks to be used in defining roles. They are more fundamental than roles because a given permission can be assigned to several different role names to create several different role definitions. Permissions are to be defined separately from the ultimate roles to which they may be assigned. However, as stated earlier, permissions should be defined in relation to one or more candidate roles. That is, permissions should be defined by considering how each permission is needed by a candidate role to accomplish a job function. Thus, in practice it is not valid to define permissions in the absence of roles, because each permission must reflect an operation on an object that has meaning within the organization. Viewed another way, in the absence of a candidate role name each object could potentially be operated on by the universe of operations possible on the object. By making permission definitions to be guided by the candidate role names that could use the permissions, the number of valid operation-object pairs will be drastically reduced. Figure 8.3 illustrates the need to define permissions in relation to candidate role names.

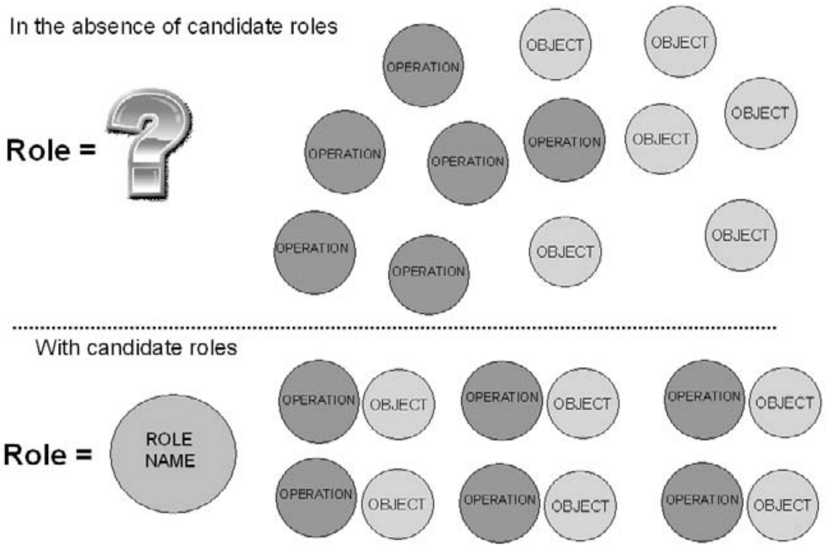


Figure 8.3 Defining permissions requires candidate role names.

When using a bottom-up approach, existing systems will yield permissions that are not initially associated with role names. This would appear to be an exception to the rule that permissions cannot be defined in the absence of role names. However, upon examining this situation we see that these existing permissions, if they are low-level IT permissions such as "read record," will still have to be generalized into abstract permissions. In this process the associated role names would be used. If the existing permissions are already abstract in nature, they were originally defined in relation to the individual users or groups that were to exercise them, because otherwise the abstract permissions would not be needed. Once abstract permissions have been derived in a bottom-up approach they will have to be generalized to the enterprise level.

As permissions are defined, they should be collected into a permissions catalog. There they will be available for use in defining roles.

Once a permissions catalog has been created, a separate activity will be conducted to assign the permissions in the catalog to the defined set of role names for the enterprise. These role names may or not be the same as the candidate roles that were used in defining the permissions. This fact may appear to be paradoxical. As stated, permissions should be available in a permissions catalog for the organization to use in creating roles.

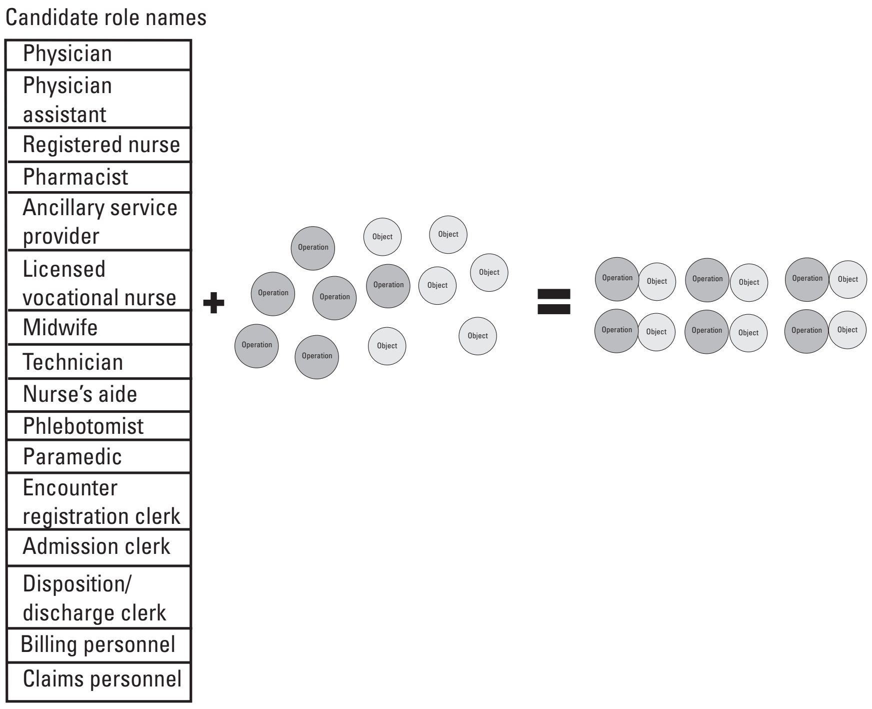
Overcoming the Paradox

The approach taken by VHA illustrates one way of accomplishing the seemingly impossible feat of using possibly different enterprise role names instead of the candidate role names. The VHA and HL7 are creating standardized permissions for healthcare. In this case there was a standard set of role names available to serve as a scaffold on which to define permissions. These are the ASTM clinical role names [1]. In practical terms, during a series of RBAC task force meetings, groups of clinical personnel (doctors, nurses, pharmacists, and laboratory personnel) met to identify permissions. The task force members were provided with the standard ASTM role names and asked to define those operations on objects that personnel referred to by the role names would typically perform. Where suitable role names were not available in the ASTM set, the group identified candidate role names to be added to the list. These were presented to ASTM for consideration. Figure 8.4 illustrates the permission identification process. Figure 8.4(a) illustrates how candidate roles are used to define permissions from operations and objects. Figure 8.4(b) illustrates how the defined permissions are combined with the particular enterprise role names to create the enterprise roles.

In one set of exercises, the group used their personal experience to identify the high-level permissions. In another set of exercises, existing scenarios and some new scenarios developed by the group were analyzed to identify operations on objects (i.e., permissions). Again, it was necessary to relate these candidate permissions to actors or candidate role names. This is how the ASTM set was used for this purpose.

Two Schools of Thought

Two schools of thought emerged from the VHA/HL7 work. One line of reasoning stated that to follow the Neumann and Strembeck methodology, the starting point should be the scenario. The process described by Neumann and Strembeck will eventually produce a permissions catalog. This approach is generally recognized to be quite labor intensive. To define a scenario it is necessary to represent an environment with a series of steps—some of them IT related and others IT independent—which would be typical in the target environment. An example of such an environment would be a patient encounter where the patient had complained of chest pains. In the scenario it is necessary to create fictitious names and to imagine the steps of the encounter. One interesting finding in developing these scenarios was that seemingly



(a)

Figure 8.4 (a, b) The permission identification process.

different scenarios can produce the same set of permissions. For example, it was found that a patient encounter for a male patient complaining of chest pain could result in the same permissions as those for a patient encounter for a patient seeking prenatal care.

The other school of thought is along the lines of "How can we short-circuit the process of developing and analyzing scenarios and achieve the same results in a shorter period of time?" The idea was that after having gone through the formal scenario creation and analysis process, the same individuals could then mentally recreate the scenario in an informal way without the need to imagine names, non-IT activities, and so on, for the scenario. Following this approach, the activities were to select a role name from the ASTM set and then mark the high-level permissions that were used by that role name. A starter set of high-level permissions was available from analysis of existing scenarios that had been developed by HL7 committees and a few newly created scenarios for the purpose. At this stage, where a permission is

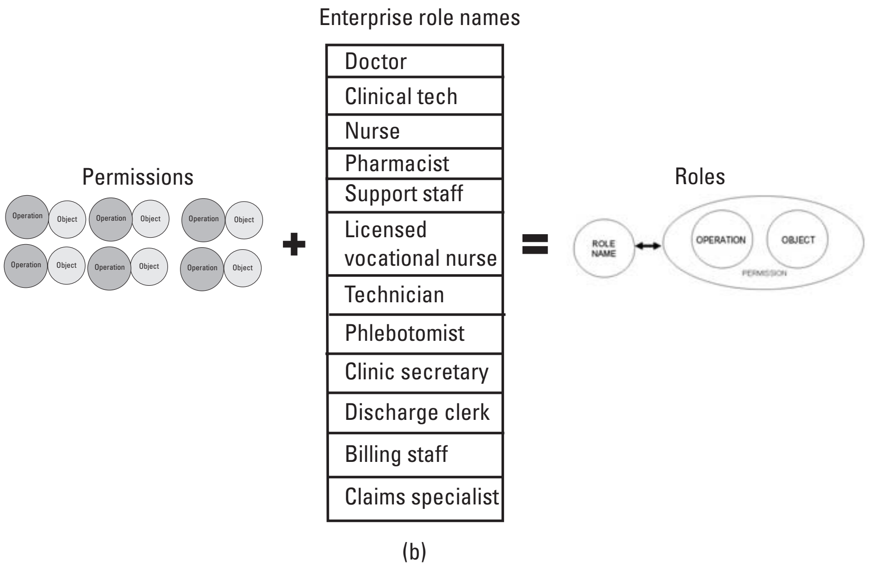


Figure 8.4 (continued)

not available to associate with a role name, the existing set is augmented using the same pattern of permission definitions previously defined. This approach using subject matter experts serves to create a catalog of high-level permissions. Figure 8.5 illustrates the scenario-based approach and the short-cut approach.

Translating High-Level Permissions into IT Permissions

The Engineering Part

When actually engineering the permissions, a general approach such as that provided here can be used. The objectives of the approach and methods by which the objectives can be met are listed in Table 8.1.

The following is an example of starting with a scenario and preparing a permission catalog. It is based on materials prepared by the VHA in conjunction with HL7.

Table 8.2 displays a summary of the content of the preceding scenario.

Using this summary of the scenario description, further analysis can be conducted by extracting the actor and action data from the summary. Only those actions using a computer system are included in the analysis.

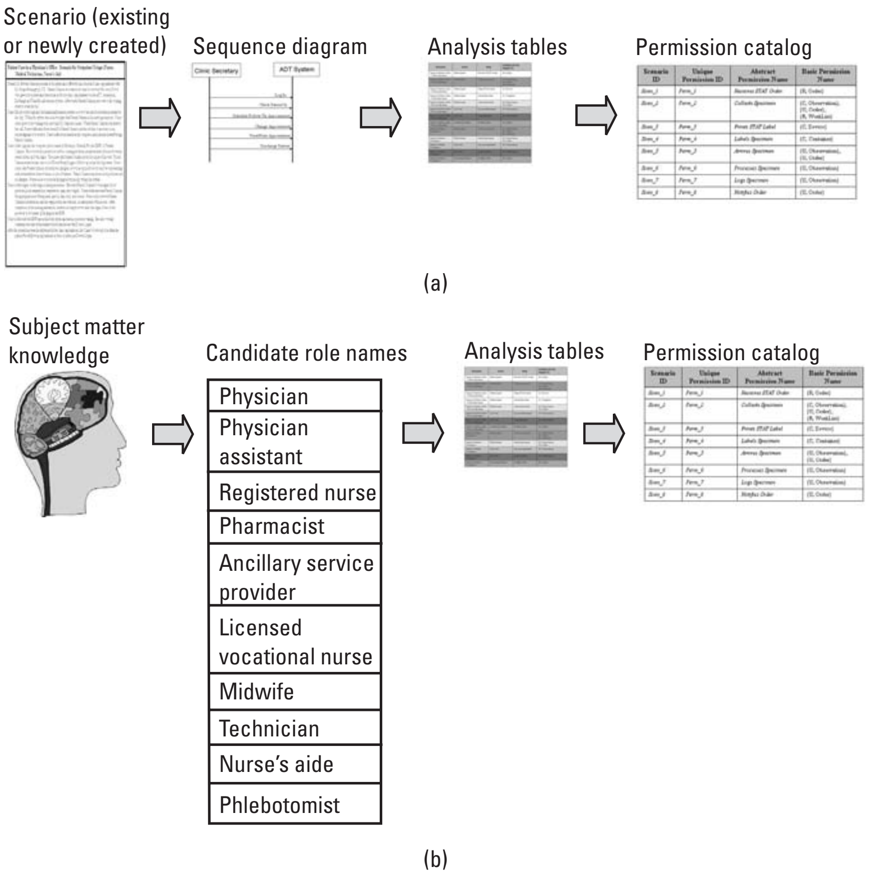


Figure 8.5 (a) Scenario-based approach and (b) shortcut approach to defining permissions. (Clip art copyright clipart.com.)

Observing the UML sequence diagrams in Figure 8.6(a, b), we note that each actor appears as a starting point for a series of transactions with an endpoint of an information system.

The informal language used in the scenario and carried over to the summary is replaced in the sequence diagrams by a transaction description that consists of a verb and an object. For example, the first action in the scenario, "Records Arrival," is represented in the sequence diagram by the transaction "Checks Patient In," that is a verb-object pair. This is done intentionally, in that the verb-object pairs are precursors to operation-object

Table 8.1
Objectives and Methods of the Permission Engineering Approach

Objective	How Accomplished
Identify operation and object pairs	Scenarios, group discussions, interviews; operations and objects must be in IT systems
Identify actors for operation-object pairs	Analysis of scenarios; these will lead ultimately to role names
Semantically cluster operation-object pairs to facilitate reduction in their number	If an operation-object pair has the same meaning as one or more other operation-object pair(s), collect these for possible renaming
Rename semantic clusters of operation-object pairs	If it is agreed that a set of operation-object pairs contain a single semantic relation, rename the pair to a common name
Cluster and rename operations	If it is agreed that a set of operations contains a single semantic entity, rename the operations in the set to a common name
Cluster and rename objects	If it is agreed that a set of objects contains a single semantic entity, rename the objects in the set to a common name
Eliminate remaining duplicate operations, objects, or actors	For those items not in a semantic cluster, resolve duplicate names to be different names
Record resulting permissions in a permission catalog	Enter resulting candidate permission names with their definitions into a catalog

pairs. Of course, the operation-object pairs are candidates for permission definitions.

Table 8.3 summarizes the high-level permissions derived from the scenario. The example IT permissions are provided to indicate how the high-level permissions could be realized in more concrete terms. The task of eliminating duplicate permissions and selecting common names for permissions that are alike semantically will be done on the high-level permissions. These are the permissions that are understood by the subject matter experts and that could be realized in many ways within IT systems.

Relating High-Level Permissions to Permissions in an IT System

To relate the high-level permissions (also called abstract permissions) to permissions in an IT system, they must be decomposed into sets of IT

Patient Care in a Physician's Office: Scenario for Outpatient Triage (Nurse, Medical Technician, Nurse's Aid)

Patient Mr. Edward Johnson presents at the physician's office for his scheduled clinic appointment with Dr. Sergei Hemingway, MD. Patient Johnson encounters the clinic's secretary Ms. Amy Dowd who greets the patient and records his arrival for the clinic appointment via the ADT (Admission, Discharge and Transfer) information system. Afterwards, Patient Johnson proceeds to the waiting room to await his call.

Nurse Mickey Adler logs into the hospital information system to review her list of scheduled patients for the day. Within the system she acknowledges that Patient Johnson is her next appointment. Nurse Adler goes to the waiting room and calls Mr. Johnson's name. When Patient Johnson responds to her call, Nurse Adler introduces herself to Patient Johnson and escorts him to an exam room, requesting him to be seated. Nurse Adler seats herself at the computer and positions herself facing Patient Johnson.

Nurse Adler logs into the computer and accesses the Electronic Medical Record (EMR) of Patient Johnson. She reviews the patient's record for existing problems, progress notes, laboratory results, recent orders, and vital signs. The nurse asks Patient Johnson about the nature of his visit. Patient Johnson states he has come to see Doctor Henry Logan to follow up on his blood pressure. Nurse Adler asks Patient Johnson about known allergies, as well as any problems he may be experiencing such as headaches, blurred vision, or loss of balance. Patient Johnson reports no such problems and no allergies. Nurse Adler records the findings electronically within the system.

Nurse Adler begins conducting a nursing assessment. She takes Patient Johnson's vital signs (blood pressure, pulse, respirations, temperature, pain, and weight). Nurse Adler assesses Patient Johnson for appropriateness of responses, speech, skin color, and lesions. Nurse Adler reviews Patient Johnson's medications and the supplies that are with him, in anticipation of his needs. After completion of the nursing assessment, medication/supply review and vital signs, Nurse Adler proceeds to document all findings in the EMR.

Nurse Adler exits the EMR and notifies the physician that his patient is waiting. She also verbally communicates any of the patient's problems/issues with Doctor Logan.

After the appointment with Doctor Logan, Nurse Adler may do the following activities for the patient:

- Enter physician-directed order(s);
- Perform or process physician's order (such as to administer a skin test or immunization);
- Refer to a consultation;
- Schedule another clinic appointment.

As part of the process of processing or screening the patient for an appointment, Nurse Adler can also:

- Check the patient in to an appointment;
- Discharge/end the patient appointment visit;
- Cancel or reschedule a patient appointment;
- Act on a series of standing orders;
- Do advanced visit planning orders (which are ordered at a previous visit for the patient);
- Send the patient for a consultation, such as to a dietitian, pharmacist, family planning councilor;
- Perform and document education;
- Perform and document health factor screenings.

After the patient has seen the physician for the clinic appointment, the Clinic's Secretary schedules the patient for a follow-up appointment in 3 months, per Doctor Logan.

operations on IT objects. These operations will include such operations as create, read, update, delete, execute, and insert. The objects will include such objects as file, record, program, and profile. The decomposition process relies on the expertise of application and system designers and analysts. Either new program logic will need to be developed or existing code will need to be identified and incorporated into the application. After the transformation from high-level to low-level permissions has been achieved, the components of the high-level permissions can be used to identify as a whole the set of high-level

Table 8.2
Summary of Content in the Scenario

Actor	Action	Reason	Using Computer System?
Patient	Present self	Appointment	No
Patient	Encounter secretary	Next step	No
Clinic Secretary	Greet patient	First step	No
Clinic Secretary	Log in to ADT System	Authentication	Yes
Clinic Secretary	Check patient in	Recognize encounter	Yes
Patient	Proceed to waiting room	Await call	No
Nurse	Log in to hospital information system	Authentication	Yes
Nurse	Review scheduled patients	Verify appointment	Yes
Nurse	Acknowledge next appointment (not an IT function)	Identify next patient	Yes
Nurse	Proceed to waiting room	Call patient	No
Nurse	Call patient	Begin healthcare process	No
Patient	Respond to call	Begin healthcare process	No
Nurse	Introduce self	Begin healthcare process	No
Nurse	Escort patient to exam room	Begin healthcare process	No
Nurse	Log in to EMR system	Authentication	Yes
Nurse	Access patient record	Continue healthcare process	Yes
Nurse	Review EMR regarding existing problems, progress notes, lab results, recent orders, and vital signs	Continue healthcare process	Yes
Nurse	Question patient regarding nature of visit	Continue healthcare process	No
Patient	State reason for visit	Continue healthcare process	No
Nurse	Question patient regarding allergies, problems	Continue healthcare process	No
Patient	Respond regarding allergies, problems	Continue healthcare process	No
Nurse	Record findings	Continue healthcare process	Yes
Nurse	Start nursing assessment	Continue healthcare process	No
Nurse	Take vital signs	Continue healthcare process	No
Nurse	Assess patient for appropriateness of responses, speech, skin color, and lesions	Continue healthcare process	No

Table 8.2 (continued)

Actor	Action	Reason	Using Computer System?
Nurse	Review medications	Continue healthcare process	Yes
Nurse	Complete nursing assessment, medication/supply review, and vital signs	Continue healthcare process	No
Nurse	Document findings in EMR	Continue healthcare process	Yes
Nurse	Exit EMR	Complete nurse process	Yes
Nurse	Notify doctor patient is waiting	Continue healthcare process	No
Nurse	Communicate patient problems/issues to doctor	Continue healthcare process	No
Nurse	Enter physician-directed order(s),	Complete healthcare process	Yes
Nurse	Perform or process physician's order (such as to administer a skin test or immunization)	Complete healthcare process	No
Nurse	Refer to a consultation	Complete healthcare process	Yes
Clinic Secretary	Schedule another clinic appointment	Complete healthcare process	Yes
Other Permitted Actions of Nurse			
Nurse	Access consultation	Begin healthcare process	Yes
Nurse	Check patient in	Recognize encounter	Yes
Nurse	Discharge patient	Complete healthcare process	Yes
Nurse	Schedule follow-up appointment	Complete healthcare process	Yes
Nurse	Cancel or reschedule appointment	Complete healthcare process	Yes
Nurse	Do advanced visit orders	Complete healthcare process	Yes
Nurse	Document education	Continue healthcare process	Yes
Nurse	Document health factor screenings	Continue healthcare process	Yes
Nurse	Access standing order	Continue healthcare process	Yes
Nurse	Enter physician-directed order	Continue healthcare process	Yes
Nurse	Enter standing order	Continue healthcare process	Yes
Nurse	Change existing order	Continue healthcare process	Yes
Nurse	Document consultation	Complete healthcare process	Yes

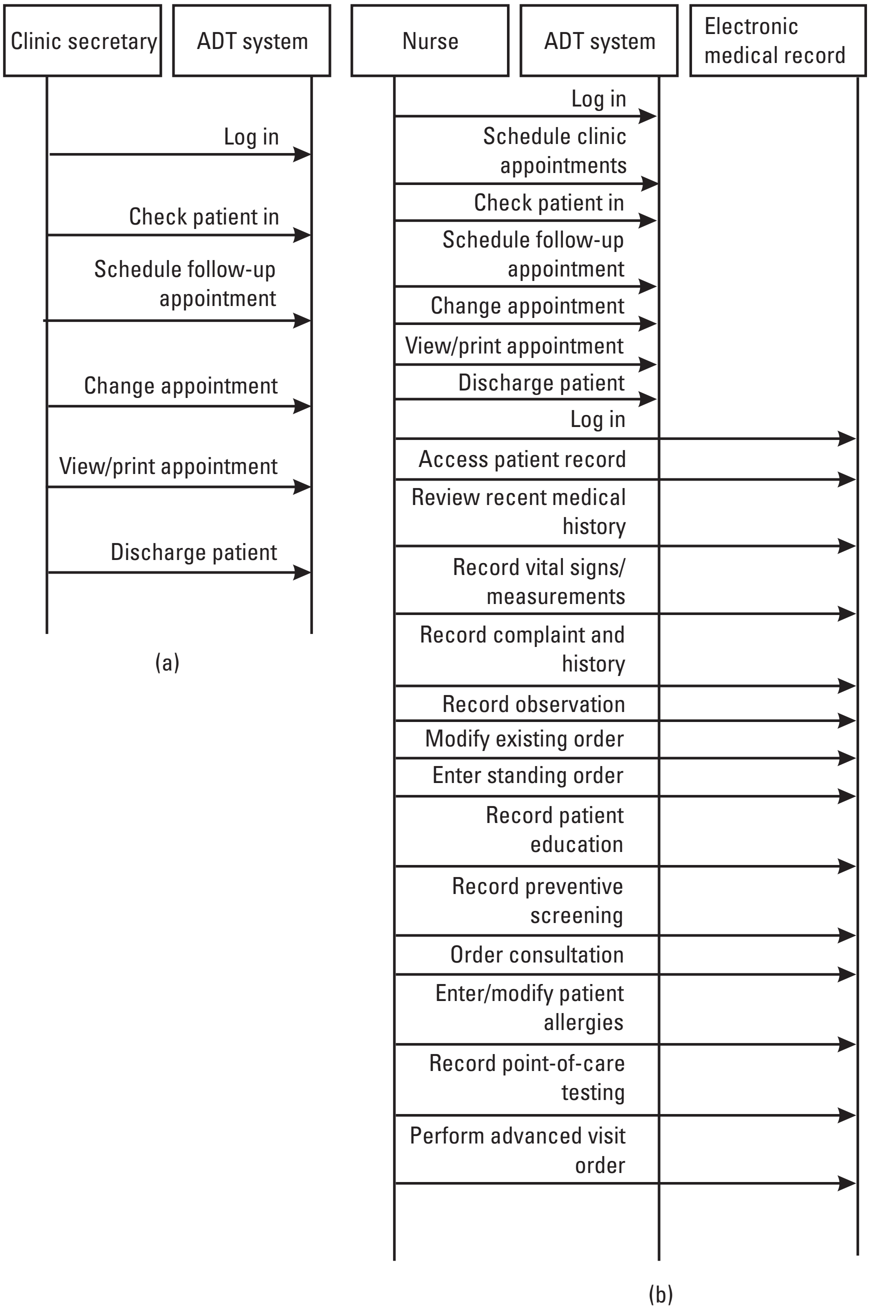


Figure 8.6 Sequence diagrams: (a) clinic secretary actor; and (b) nurse actor.

Table 8.3
Actors, High-Level Candidate Permissions, and Example IT Permissions

Actor	High-Level Permission	Example IT Permission*
Clinic secretary	Check in patient	{C, Patient Encounter}
Clinic secretary	Discharge patient	{U, Patient Encounter}, {E, Patient Encounter}
Clinic secretary	Schedules appointment	{C, Patient Encounter}, {U, Patient Encounter}
Clinic secretary	Change appointment	{U, Patient Encounter}, {D, Patient Encounter}
Nurse	Schedule clinic appointment	{C, Patient Encounter}
Nurse	Print clinic appointment	{R, Patient Encounter}
Nurse	Checks patient "in"	{C, Patient Encounter}
Nurse	Discharges/ends patient appointment visit	{U, Patient Encounter}, {E, Patient Encounter}
Nurse	Schedules follow-up appointment(s)	{C, Patient Encounter}, {U, Patient Encounter}
Nurse	Cancels/reschedules patient appointment(s)	{U, Patient Encounter}, {D, Patient Encounter}
Nurse	Accesses patient record	{R, Patient Encounter}
Nurse	Reviews recent medical history	{R, Patient Encounter}
Nurse	Performs/records vital signs and patient measurements	{C, Patient Encounter}, {U, Patient Encounter}
Nurse	Records chief complaint and medical history	{C, Patient Encounter} {U, Patient Encounter}
Nurse	Records observations	{C, Patient Encounter}, {U, Patient Encounter}
Nurse	Modifies/cancels existing order(s)	{C, Patient Encounter}, {U, Patient Encounter} {C, Procedure}, {U, Procedure}
Nurse	Enters standing orders PRN	{C, Patient Encounter}, {U, Patient Encounter}, {C, Procedure}, {U, Procedure}

Table 8.3 (continued)

Actor	High-Level Permission	Example IT Permission*
Nurse	Performs/records patient education	{C, Patient Encounter} {U, Patient Encounter} {C, Procedure} {U, Procedure}
Nurse	Performs/records preventive screening (alcohol, STD, tobacco)	{C, Patient Encounter}, {U, Patient Encounter}, {C, Procedure}, {U, Procedure}
Nurse	Enter consultation(s) PRN	{C, Patient Encounter}, {U, Patient Encounter} {C, Procedure}, {U, Procedure}
Nurse	Enters/modifies/deletes patient allergies	{C, Patient Encounter}, {U, Patient Encounter}
Nurse	Records point-of-care results	{C, Patient Encounter}, {U, Patient Encounter}, {C, Observation}, {U, Observation}

*C = create; R = read; U = update; D = delete; E = execute.

and subordinate low-level permissions. That is, the hierarchy of permissions from high to low level can be identified and handled by the highest-level permission. Figure 8.7 illustrates the translation of high-level permissions to low-level IT permissions.

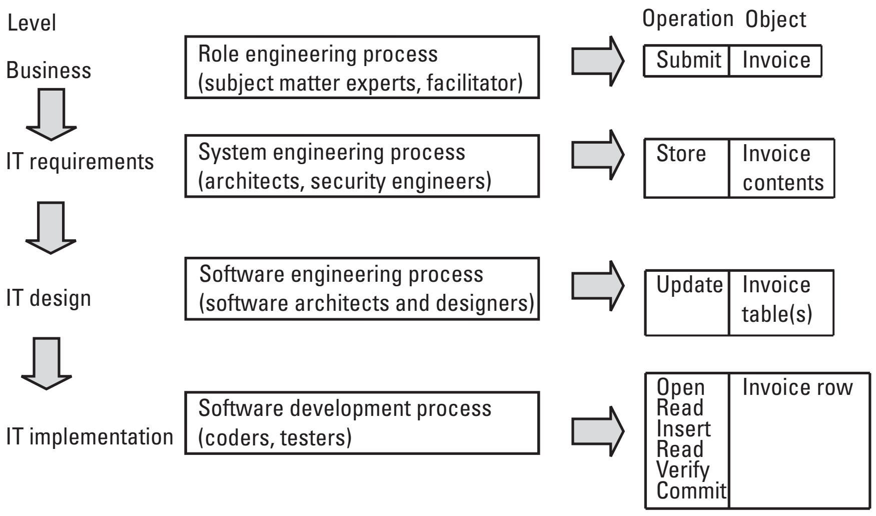

Level
Business
IT requirements
IT design
IT implementation
Role engineering process (subject matter experts, facilitator)
System engineering process (architects, security engineers)
Software engineering process (software architects and designers)
Software development process (coders, testers)
Operation
Object
Submit
Invoice
Store
Invoice contents
Update
Invoice table(s)
Open Read Insert Read Verify Commit
Invoice row

Figure 8.7 Translating high-level permissions to IT permissions.

Reference

[1] American Society for Testing and Materials (ASTM), *E 1986–98 Standard Guide for Information Access Privileges to Health Information*, November 1998, http://www.astm.org.

9

Tools That Can Be Used to Assist the Role Engineering Process

As is the case with similar efforts to elicit, analyze, and record requirements, software tools can facilitate the role engineering process. Tools have the ability to store and process data as it is gathered and analyzed. Generic tools, such as spreadsheet and database products, are the minimum level of tools that should be available to a role engineering effort. Beyond these generic types of tools, more specialized role engineering tools are available in the marketplace. Use of these can in some cases promote efficiency and possibly quality in the results of a role engineering effort. However, it should not be assumed that any more specialized tools are required to achieve success in the effort; selection and adoption of other tools will depend on the specific project and environment. It is recommended that additional tools at least be reviewed and considered for adoption. This requires identification of suitable products and then evaluating their expected return on investment.

Potential Benefits of Role Engineering Tools

The potential benefits of specialized role engineering tools can be related to the staff available to the role engineering effort and their level of role engineering expertise. To a certain extent, the correlation will be positive: for staff with lower expertise the need would be for more basic tools, and for staff with higher expertise the need would be for more sophisticated tools. This positive correlation is based on the requirement for more sophisticated knowledge and experience to capitalize on the capabilities of the tools. A less experienced staff member would be expected to possess fewer possibilities for recognizing the benefits of the tools and applying the tools to the many role engineering tasks requiring attention.

A negative correlation between the level of expertise of available staff and the value of more sophisticated role engineering tools also exists. If the specialized tools have significant capabilities for enforcing a role engineering process in a helpful way, those tools could assist less experienced staff with following the process and performing the tasks needed to define roles and their structures and constraints. Of course, the staff would need sufficient training to understand the tools and to achieve the available benefits of using them. An experienced mentor can help with this process of learning and acclimation. Such a mentor would also be valuable in checking decisions and results produced by the staff. More experienced team members should be aware of the possibility of less experienced staff overly relying on tools without sufficient application of professional judgment.

What Tools Can Do

Special-purpose role engineering tools are capable of performing a variety of functions that may or not be relevant to a given role engineering effort. The

Table 9.1
Categories of Role Engineering Functions and Tools

Role Engineering Function	Tool Category
Defining permissions	Role discovery (role mining)
Relating permissions to roles	Role discovery (role mining)
	Analytical
Optimizing role structures	Modeling
Reflecting workflows	Modeling
Organizing role definitions in a repository	Repositories
Designing the role structure	Modeling
Modeling role structures	Modeling
Testing the role definitions	Analytical
	Role enforcement (secondary significance to role engineering)

tools can provide benefits in several areas. Table 9.1 summarizes the functions performed by available tools.

Deciding Whether Tools Are Needed

Based on the experience of role engineering efforts such as that of the Veterans Health Administration, we can advise that specialized role engineering tools are not necessarily required. It is perfectly acceptable to begin a role engineering effort using readily available generic tools such as spreadsheet and database products. Other generic tools might include text editors, configuration management products, and repository products. These products themselves contain software functions that can aid in the definition of roles, analyzing their characteristics, recording the results of these efforts, and reporting on this information. Table 9.2 summarizes some generic tools and how they may be used for role engineering. They are of a general nature and permit their files to be imported into other, more specialized tools at a later time if desired.

We suggest that a good approach may be to start simple and use basic tools such as databases, spreadsheets, and other tools such as those shown in Table 9.2. When time and resources permit, it will be possible to consider whether or not specialized tools could facilitate the role engineering process.

Table 9.2
Generic Tools for Role Engineering

Tool	**Possible Uses**
Spreadsheet	Manage matrices of candidate roles versus abstract permissions; sort statements of abstract permissions for clustering, merging, and splitting
Database	Record and maintain role, permission, and constraint definitions
Text editor	Review and modify role definitions
Word processor	Review and modify role definitions; record use case and similar data
Diagram drawing tool	Prepare diagrams such as UML sequence diagrams
Configuration management	Control updates to components of the role definition process
Repository	Record and maintain role, permission, and constraint definitions as well as publish interim and final results to interested parties and system components

For example, it may be the case that collaboration tools could be employed to boost productivity. Another need for specialized tools may arise where the quantity of data is too massive or the complexity of role definitions is too great for some of the generic tools. On the other hand, it should be borne in mind that some of the more specialized tools will have significant acquisition and maintenance costs as well as training requirements for users.

While the role engineering effort is underway, or after it has been concluded, an effort can be undertaken to identify, evaluate, and possibly adopt tools whose use appears to be promising. In the meantime, while using basic tools, care should be taken to record the data in a manner that facilitates migration of that data to specialized tools in the future. Figure 9.1 illustrates the process of identifying a potential need for a specialized role engineering tool and leading to the implementation of the tool.

It is important not to let the use of tools interfere with the fundamental job of defining roles and their structures and constraints. As noted, generic products can go a long way in supporting the role engineering effort. It would be a mistake to pay more attention to tools than to the needed results of role engineering.

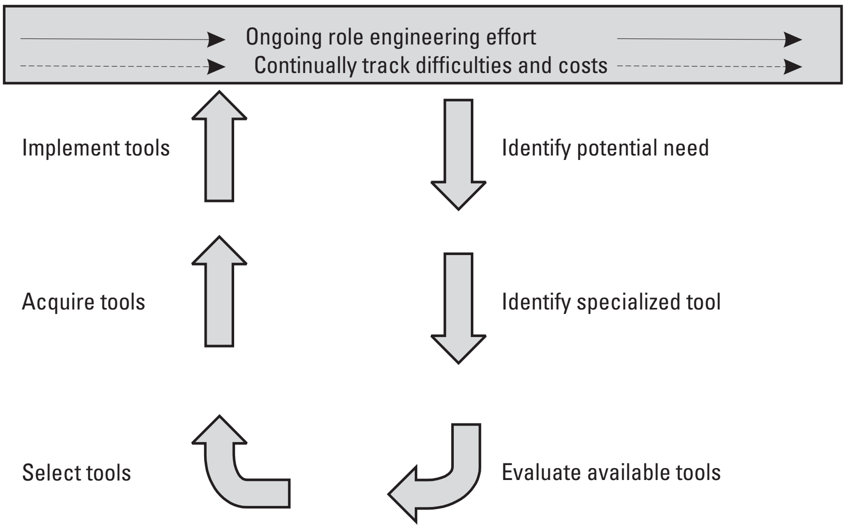

Ongoing role engineering effort
Continually track difficulties and costs
Implement tools
Identify potential need
Acquire tools
Identify specialized tool
Select tools
Evaluate available tools

Figure 9.1 Identifying the need for specialized role engineering tools.

What Tools Cannot Do

While tools can be helpful and can promote efficiency and multiply analytical power, they are secondary to the diligent application of the analytical and logical powers of the staff conducting role engineering activities. Thus they are not a substitute for the powers of human reason. By themselves they cannot create information. The adage of "garbage in, garbage out" applies to the use of tools for role engineering as in other endeavors. Tools, therefore, should be used judiciously to avoid establishing false hopes for them and possibly bringing discredit on a role engineering effort by misapplying them or overly relying on them.

Tool Selection Criteria

Selection of role engineering tools in many ways is no different from the selection of other products by an enterprise. Selection criteria are provided below. We note that it will be difficult to determine values for some of the criteria before adequate experience has been obtained from practical role engineering experience. In particular, return on investment will be an unknown until experience has been obtained. With an eye to obtaining this experience, an effort will need to be made during role engineering activities to identify and record data on the costs and benefits of having or not having specialized tools. Only then will it be feasible to analyze return on investment of tools that are under consideration. Therefore, we recommend that a set of metrics be identified at the inception of the role engineering effort and that data be collected for these metrics. The metrics would measure efficiency, effectiveness, and costs. Armed with this data, analysts can measure the potential return on investment of improved role engineering tools in support of the decision-making process.

Table 9.3 summarizes tool selection criteria that will be relevant to specialized role engineering tools.

Cost-Benefit Analysis

A cost-benefit analysis should be conducted to determine the advisability of acquiring a new tool. Some of the basic tasks for conducting this analysis are provided in Table 9.4.

Table 9.3
Tool Selection Criteria

Tool Selection Criterion	**Relevance**
Cost	Acquisition and life-cycle costs assist in the cost effectiveness decision
Return on investment	Potential benefits as compared with not acquiring a tool
Scalability	Potential for enterprise-wide usage and sharing of results
Flexibility	Potential for using the tool in various environments and for various purposes
Support	Availability of technical support and product updates
Functionality	Available functions provided by the tool
Interoperability	Degree and type of integration with related tools (e.g., ERP/HR systems, business process modeling, requirements management)
Ease of use	Level of tool experience required to obtain the tool's benefits
Training requirements	Level of training required to obtain the tool's benefits

Some Available Tools

Some of the leading tool vendors and their tools are listed in Table 9.5. No particular endorsement or ranking is implied by this list of tool suppliers.

Table 9.6 identifies some of the available tools that support role engineering activities.

Tools Summary

Role engineering can be conducted successfully without the use of specialized tools. We recommend beginning an enterprise role engineering effort using only generic tools such as spreadsheets and databases. As role engineering progresses, an effort should be made to collect data on recognized difficulties. It may turn out that specialized tools could play a role in alleviating these difficulties. Without this data it will be difficult to analyze at a later date the potential return on investment of a tool.

Where it has been determined that a new tool could provide value to an enterprise's role engineering efforts, a set of candidate tools should be assembled for further consideration. Alternatives for acquisition of the tools should be identified, and the properties of the tools should be assembled to support

Table 9.4
Cost-Benefit Analysis

Task	**Description**
Determine projected costs without the tool	Identify actual or relative costs of current or previous projects focusing on activities where the tool would be used
Determine potential benefits	Identify savings in costs or time plus improvements in expected results
Establish alternatives	Identify options for acquiring the product such as lease, purchase, Web-based, locally installed, existing licenses
Apply trade-off analysis of alternatives	Weigh alternative according to costs, effectiveness, compatibility with existing products, training requirements, and other relevant criteria

a side-by-side analysis of alternatives. Comparison among the tools will be structured according a set of evaluation criteria, which should include the following:

- Cost;
- Return on investment;
- Scalability;
- Flexibility;
- Support;
- Functionality;
- Interoperability;
- Ease of use;
- Training requirements.

Following this evaluation, decisions will be made for acquiring or not acquiring any specialized role engineering tools. The acquisition process will require planning for purchase (or other method) and implementation. Necessary justifications for funding can use the data previously developed in the evaluation process. In particular, the justifications can focus on the evaluation criteria and the values developed in relation to these criteria.

Table 9.5
Tool Vendors and Their Tools

Organization	Approach	Products	Contact
Prodigen http://prodigen.com	Bottom-up	Contouring Engine	info@prodigen.com
Eurekify http://www.eurekify.com	Primarily bottom-up	Sage Enterprise Role Management	info@eurekify.com
Bridgestream http://www.bridgestream.com	Primarily top-down	SmartRoles	sales@bridgestream.com
BHOLD http://www.bholdcompany.com/	Bottom-up	BHOLD MANAGER BHOLD MODELER BHOLD AUDITOR	support@bholdcompany.com
Vaau www.vaau.com	Both	RBACx	info@vaau.com
Beta Systems http://ww2.betasystems.com/en/index.html	Bottom-up	SAM Jupiter Role Miner	info@betasystems.com
Generic http://www.generic.se/	Top-down	Vaktis	info@generic.se
IBM http://www-306.ibm.com/software/tivoli/products/identity-mgr/	Bottom-up	IBM Tivoli Identity Manager	
Courion http://www.courion.com/	Both	RoleCourier	info@courion.com
SecurIT http://www.secureit.com/	Bottom-up	Tivoli Identity Manager Eurekify Sage Enterprise Role Management	
Siemens http://enterprise.usa.siemens.com/products/solutions/security/idmanage.html	Bottom-up	HiPath DirX	
Sun Neogent http://www.neogent.com/	Bottom-up	Bi Level RBAC	info@neogent.com

Table 9.6
Tools that Support Role Engineering Activities

Role Engineering Activity	**Available Tools**
Defining permissions	Spreadsheet/database
	Eurekify Sage Enterprise Role Manager
	Beta Systems SAM Jupiter Role Miner
	BHOLD MODELER
	Prodigen Contouring Engine
	Generic Integration
	Vaktis
	Siemens HiPath DirX
	Vaau RBACx
	Bridgestream SmartRoles
	Courion RoleCourier
Relating permissions to roles	Spreadsheet/database
	BHOLD MANAGER
	Sun Neogent Bi Level RBAC
	Beta Systems SAM Jupiter Role Miner
	Prodigen Contouring Engine
	Generic Integration
	Vaktis
	Siemens HiPath DirX
	Vaau RBACx
	Courion RoleCourier
	Bridgestream SmartRoles
Optimizing role structures	Spreadsheet/database
	Eurekify Sage Enterprise Role Manager
	Eurekify Sage Enterprise Role Manager (ERM) for IBM Tivoli Identity Manager
	Courion RoleCourier
Reflecting workflows	Modeling tool
	BHOLD MANAGER
	BHOLD MODELER
	Bridgestream SmartRoles
	IBM Tivoli Identity Manager
	Siemens HiPath DirX

Table 9.6 (continued)

Role Engineering Activity	**Available Tools**
Organizing role definitions in a repository	Repository product
	Eurekify Sage Enterprise Role Manager (ERM) for IBM Tivoli Identity Manager
	Siemens HiPath DirX
	Sun Neogent Bi Level RBAC
	Vaau RBACx
	Bridgestream SmartRoles
Designing the role structure	Design tool
	Eurekify Sage Enterprise Role Manager

10

Putting It All Together: The Role Formation Process

At this point in the role engineering process, we should have available a permissions catalog, a set of candidate (scaffold) role names associated with the permissions, a set of enterprise role names to replace or supplement the candidate role names, an identification of constraints, and perhaps some defined role hierarchies.

Combining the Ingredients

Once the hard work has been accomplished of determining the access control policy to be enforced with RBAC, the role names, the high-level permissions, the low-level permissions, and the constraints, the next step is to combine these ingredients to form roles. Some of these steps have been discussed earlier. Here we make a synthesis of all the components with the goal of achieving a result that is valid across the enterprise.

Workflows

Users in a workplace situation often do not use individual system capabilities in isolation. The more likely scenario is that they execute sequences of commands or transactions during the course of achieving an automated result. A related set of these commands or tasks is known as a workflow. A workflow is simply a set of tasks to be performed by one or more users to accomplish a desired result. What unifies these sets of tasks into a workflow is the overall result that is accomplished. Examples of workflow names are as follows:

- Create purchase order;
- Fulfill purchase order;
- Process new customer;
- Document treatment plan.

Each of these workflows will contain the set of tasks that must be performed to fulfill the goals of the workflow.

The relationship of RBAC to workflows is addressed by Ferraiolo et al. in [1], where it is pointed out that operational context, such as execution history, can be involved in access control decisions. Thus it may be advisable in some instances to relate the role definitions to the components of a workflow, possibly via a workflow management system (WFMS). This should not affect the central role definitions (role name-permission), but the relationships to workflows will appear as constraints on roles. Where roles are to be related to workflows, these relationships can be included in the role repository, which is discussed later in this chapter.

Individual users of an IT system very often participate in workflows according to their assigned job functions and responsibilities. A workflow may or may not be supported by a WFMS. In either case, role definitions should be developed within a workflow context where a workflow is present.

According to Kang et al. [2], "From an organizational-level access control point of view, a workflow is an ordinary application program acting on behalf of users." This observation is referring to a workflow management system that schedules and controls the flow of work elements among individual workers within an enterprise. These individuals constitute users who are assigned to functional roles. We note that a user who is authorized to engage in a given workflow must have sufficient permissions available to perform the tasks of the workflow. These permissions will collectively be assigned to the functional roles within each workflow.

A workflow is a unifying force for defining and organizing the roles needed by users to perform job functions in IT systems. Thus, in synthesizing roles for the enterprise it is useful to examine the workflows as a context within which role definitions will exist. First, structural roles will be defined to control access to the enterprise's workflows themselves. This entails the definition of permissions that represent the entry to the workflows and the assignment of these permissions to structural roles. Next, sets of functional roles implicitly exist within workflows. They must be identified and their permissions must be documented. Once this process is complete, these sets of roles existing in workflows must be engineered to be mutually compatible and to contain the permissions that the workflow requires. By compatible we mean that the roles within a workflow contain the permissions needed to execute the tasks within the workflow and that the roles do not mutually overlap in their sets of permissions. This still allows for roles that are nested within a hierarchy that do not share permissions.

Relating Permissions to Roles

When defining a role we first consider relating permissions to roles. At this point, any identified permission-role constraint definitions must be consulted and implemented. For each available role name, the corresponding permissions are then assigned. The candidate (scaffold) role name-permission matrix should have guided this assignment of permissions to role names. Once the permissions have been assigned to the roles, a test should be conducted to ensure that the access control policy is properly enforced by the role. This will close the loop and ensure that the role name, permission, and constraint specifications have been accurately preserved. If they are not, the permissions may have to be reengineered or possible additional permissions must be defined and designed. Hopefully, these will be minimal. At the end of this process a set of roles will be available. It is recommended that a testing process involving the entire set of available roles be conducted. This will facilitate adjustments of permission assignment to roles. Where appropriate, the role set may be adjusted such that permissions are redistributed among the role names. Or it may be required to subdivide permissions to better match the role names to which they are to be assigned.

There is a need to subdivide a permission where part of a transaction is performed by one role and part is performed by another role. For example, a permission "Modify Customer Record" may have been defined initially with the notion that a single role "Customer Representative" would be able to read a customer record and modify that record to assist a customer. If it is determined that the "Customer Representative" permissions will actually be realized by two roles, for example, "Customer Screener" and "Customer Service Representative," these could be assigned the split permissions of

"Identify Customer Need" and "Adjust Customer Record." This example is summarized in Table 10.1.

Role Hierarchies

Part of this formation of roles may be the design of hierarchies among the roles. This entails designing the role structure or hierarchies of roles with inheritance of permissions. It was recommended during the role formation process that any creation of hierarchies be postponed to a later stage of the formation process. Now we are at the stage where hierarchies should be considered. To some extent the use of hierarchies is considered as an optimization process on a set of roles that are workable if not efficient. Efficiency here means that when a role is subdivided into junior roles it becomes possible to be more precise in granting sets of permissions to users. This use of hierarchies of roles can achieve a condition of least privilege. This will occur when a role with a relatively large number of permissions, which may be too many for a user, is subdivided into roles that more closely match the permission sets absolutely required for particular roles. For this purpose the creation of hierarchies may have been addressed earlier in the role creation process. The introduction of hierarchies will involve the decomposition of unoptimized or overly permissive roles, with the permissions of the original roles parceled out to more junior roles in the hierarchy.

Previously conducted trade-off analysis (see Chapter 7) should be consulted to provide guidance on if and where role hierarchies will be employed. Some general principles on why a hierarchy should be used are provided, followed by some principles for defining hierarchies.

The following are reasons why hierarchies among roles should be used:

Table 10.1
Example of Splitting a Permission

Role Before Being Split	Permission Before Splitting Role	Roles After Splitting	Permissions After Splitting Role
Customer Representative	Modify Customer Record	Customer Screener	Identify Customer Need
		Customer Service Representative	Adjust Customer Record

1. Candidate roles include roles with large numbers of permissions, causing difficulty in understanding what access the roles actually provide.
2. Candidate roles include roles with large numbers of permissions that would not conceivably be assigned to a single user.
3. One or more candidate roles has some permissions that are considered to be unrelated.
4. Sets of permissions in several candidate roles seem to repeat among the roles.

These conditions indicate that the pertinent candidate roles be subdivided to form junior roles that contain fewer permissions than the original role from which they have been created.

The following are principles for defining hierarchies among roles:

1. Subdividing senior roles into junior roles should be carried out judiciously to avoid excessive complexity in the resulting role structure.
2. Junior roles created from senior roles should be assignable to users—that is, not merely a set of permissions that does not relate to a job function for the user.
3. Junior roles should be assigned meaningful names that reflect job function.
4. Candidate hierarchies should be thoroughly tested to ensure the correct implementation of the access control policy. It is possible that inheritance of the permissions of multiple junior roles can produce undesired combinations of permissions in senior roles. This possibility must be avoided.

Thus, role hierarchies are formed by dividing a role into subroles that take on some of the permissions of the senior role. These permissions will still be available to the senior role through inheritance. The judicious subdivision of roles into junior roles requires the application of several criteria. These criteria should guide the subdivision of roles into junior roles.

First, the junior roles should be defined such that a single user could be assigned to the role and not have an excessive (according to the access control policy) number of permissions available to him. If a newly created junior role does contain an excessive number of permissions, it should be further subdivided just as the original role was subdivided.

The junior roles should have coherent sets of permissions assigned to them according to the same criteria of participation in a workflow as the original senior role. Where sets of permissions in several candidate roles seemed to repeat among the roles, it is a judgment call whether or not these sets should be assigned to junior roles that combine these sets. In some cases these roles will be assigned to users according to the access control policy, and in some cases these permission bundles (sometimes called permission-only roles) could be defined simply for convenience in assigning permission clusters to other roles to which users will be assigned.

Reflecting Constraints

At this point in the process, the constraints between permissions that were previously defined (see Chapter 8) are made explicit. The actual implementation in systems will be left to a later stage and will require code design at some level. Constraints on permissions will be global, such that they do not require run-time information to be enforced.

Now any constraints between roles that were previously defined (see Chapter 7) are made explicit. As with constraints among permissions, the actual implementation in systems will be left to a later stage and will require code design at some level. Static separation of duty constraints will actually be implemented by the user assignment tool, which will prohibit a pair of roles from being assigned to the same user. Dynamic separation of duty or other dynamic role-role constraints will be implemented in the run-time system.

To summarize, Table 10.2 presents the necessary ingredients, or components, to be combined to design roles with their possible hierarchical relationships and constraints.

Process for Role Formation

We present a step-by-step process for forming roles from the ingredients that have been previously defined. This process is intended to be more of an illustration of the composition of roles from the ingredients than an algorithm to be executed in a rote manner.

Table 10.2
Role Components

Component	Definition	Comment
Users	Identified individuals who are assigned to roles	For the most part, users are not relevant to an access control policy except when there is a static separation of duty or similar constraint involving user assignment to roles
Permissions	Operation-object pair	High level (abstract) or lower level (system dependent)
Candidate role names	Temporary role names	Used as a "scaffold" to support the definition of permissions
Enterprise role names	Permanent role names	Used to represent enterprise job functions
Constraints	Permission-permission, permission-role, role-role, user-role	Attributes of permissions, roles, and users that prohibit certain combinations of these
Hierarchies	Senior and junior roles with permissions distributed among them	Named supersets and subsets of permissions with inheritance of permissions by senior roles from junior roles
Access control policy	Object, operations permitted on objects, job functions permitted to operate on the objects, and constraints prohibiting certain combinations of permissions, role names, and users	The basis for defining roles and some user assignments to roles

We start with these ingredients:

1. Users (if part of access control policy);
2. Permissions (operation-object pairs);
3. Role names;
4. Constraints;
5. Hierarchies (possibly);
6. Access control policy.

These ingredients appear in the following process descriptions.

Main Process

1. Start.
2. Check permission against access control policy. Are all operations on objects reflected in permissions?
    a. Yes—go to 3.
    b. No—identify objects and operations to complete access control policy coverage—go to 2.
3. Are enterprise roles being used (vice candidate roles or scaffold roles)?
    a. Yes—go to 4.
    b. No—Substitute enterprise roles for candidate roles—go to 4.
4. Check role names against access control policy. Are all assignments of role names to permissions reflected?
    a. Yes—go to 5.
    b. No—add role names and assign them to permissions—go to 4.
5. Have constraints been identified?
    a. Yes—invoke constraints process.
6. Have hierarchies been identified?
    a. Yes—invoke hierarchies process.
7. Insert role definition into repository.
8. Done.

Constraints Process

1. Have permission-permission constraints been identified?
    a. Yes—check role definitions for permission-permission constraints. Fix and go to Main Process 2.
2. Have role-role constraints been identified?
    a. Yes—check role definitions for role-role constraints. Fix and go to Main Process 2.
3. Have role-permission constraints been identified?
    a. Yes—check for role definitions role-permission conflicts. Fix and go to Main Process 2.
4. Have user-role constraints been identified?

   a. Yes—ensure management tools correctly enforce user-role constraints.
5. Done.

Hierarchies Process

1. Select next senior role. Last senior role?
   a. Yes—go to 4.
2. Select corresponding junior role(s).
3. Test inherited permissions against access control policy. Result satisfactory?
   a. Yes—go to 1.
   b. No—Fix and go to 3.
4. Done.

Testing Roles Against Access Control Policy

Tests should be conducted to ensure that the access control policy is properly enforced by the role definitions. To perform these tests it is convenient to have both the access control policy and the role definitions represented in tabular form. Figure 10.1 illustrates a simple role structure. Table 10.3 provides a corresponding tabular representation of the roles and constraints depicted in the figure.

Table 10.4 represents an access control policy. The columns in the table are arranged in the same order as those in the tabular representation of the roles. The role and permission names in the policy table have been associated with the symbolic names shown in Table 10.3.

Note that there is a discrepancy between the effective permissions of role R-01, in that the inherited permission P-04 does not appear in the access control policy.

For purposes of checking role definitions against access control policy, the center of focus is the role name. While the center of focus when the roles were being defined was the object and the operations permitted on it by actors or candidate role names, after the role definitions have been completed they take on an identity of their own. Thus the focus is on role names for the checking process.

The process for checking role definitions against the access control policy is as follows:

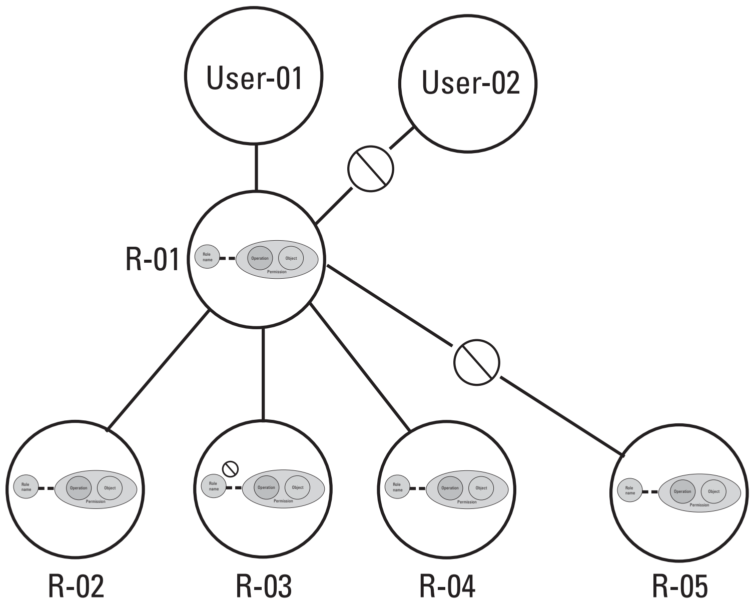


Figure 10.1 Simple role structure.

Construct a table that models the role structure and a table that summarizes the access control policy. Examples are provided in Tables 10.3 and 10.4.

Main Process

1. Start.
2. Select a role in the role catalog and the corresponding role in the access control policy. Do the assigned and inherited permissions for the role match those in the access control policy?
   a. Yes—go to 3.
   b. No—add role name with permission set to discrepancy list and go to 2.
3. Do the constraints for the role match those in the access control policy?
   a. Yes—go to 4.
   b. No—add role name with constraints to discrepancy list and go to 2.
4. Done.

Table 10.3
Tabular Representation of Roles

Modeled Role Structure								
Name	**Assigned Permission**	**Inherited Permission**	**Junior Role**	**Senior Role**	**Permission-Permission Constraint**	**Role-Permission Constraint**	**User-Role Constraint**	**Role-Role Constraint**
R-01	P-01	P-02, P-03, P-04	R-02, R-03, R-04				USER-02	R-05
R-02	P-02, P-04			R-01				
R-03	P-02, P-03			R-01		X		
R-04	P-04			R-01	X			
R-05	P-01, P02							R-01

Table 10.4
Access Control Policy

Role Name	Permission	Operation	Object	Constraint
R-01	P-01, P-02, P-03	Op-01	Ob-01	U-R: USER-02
		Op-02	Ob-02	R-R: R-05
		Op-02	Ob-03	
R-02	P-02, P-04	Op-02	Ob-02	
		Op-02	Ob-04	
R-03	P-02, P-03	Op-02	Ob-02	R-P: P-04
		Op-02	Ob-03	
R-04	P-04	Op-02	Ob-04	P-P: P-02/P-04
R-05	P-01, P-02	Op-01	Ob-01	R-R: R-01
		Op-02	Ob-02	

Op = operation; Ob = object; U-R = user-role; R-R = role-role; R-P = role-permission; P-P = permission-permission.

Organizing Role Definitions in a Repository

When role definitions are complete, they must be recorded for subsequent use and maintenance. It is not necessary to use specialized software products to hold the definitions, but it will prove useful to use a product with sufficient capability and flexibility to support required maintenance and access activities. Possibly the least common denominator of products would be a relational database with required application software. Some useful characteristics of a role repository and associated software functionality are listed in Table 10.5.

Once a suitable repository has been established, it will be necessary to populate it with role definitions. A suggested process for populating the role repository is provided here.

Each role representation in the repository should have an independent entry. The columns illustrated in Table 10.3 should be included. In addition to these columns, a narrative description should accompany the role entry. This narrative description of the role should be geared to the administrative

Table 10.5
Criteria for a Defining and Creating a Role Repository

Interface for single item and batch additions
Textual search on any RBAC component
Keyword search on any RBAC component
Support for cross-references among RBAC components
Flexible reporting capability
Export of role definitions in useful formats, for example, comma delimited, XML
Audit trail of modifications to the repository

users of the repository who will be verifying the roles assigned to functional users against those users' job function and responsibilities. Establishment of cross-references will assist role maintenance personnel in reviewing role definitions and relating these to one another. For example, it should be possible to select a role definition and to relate to it all other roles containing the selected role's permissions. Similarly, it should be possible to select an object within a permission definition and to identify all other permissions that contain that object in their definition.

With a populated role repository available, role assignment and role maintenance personnel will access the repository for various reasons, including selecting roles for assignment to users and maintaining the validity of the role definitions. Suggested processes for using the role repository are provided in Tables 10.6 and 10.7.

With the completion of these role creation, documentation, and preliminary maintenance activities, a role structure will be available for use. Figure 10.2 illustrates all of the activities involved in constructing role

Table 10.6
Repository Use by Role Assignment Personnel

Determine user need for role assignment
Locate role name in repository
Verify user need with available roles
Perform any other verification required by organization policy
Assign user to structural roles
Assign user to functional roles
Make recommendations for new role definitions

Table 10.7
Repository Use by Role Maintenance Personnel

Verify that all currently authorized roles are included in the repository
Verify role definitions against current access control policy
Verify that repository functionality is working correctly
Insert new or modified role definitions
Remove invalid role definitions
Run inter-role consistency checks

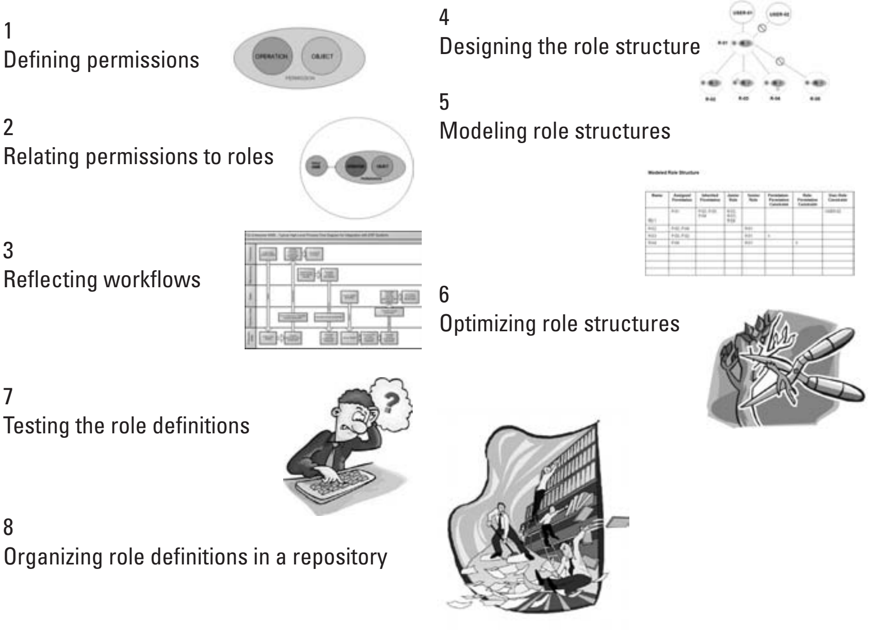

1
Defining permissions
OPERATION
OBJECT
2
Relating permissions to roles
3
Reflecting workflows
4
Designing the role structure
5
Modeling role structures
6
Optimizing role structures
7
Testing the role definitions
8
Organizing role definitions in a repository

Figure 10.2 Constructing role definitions and associated role structures. (Clip art copyright clipart.com.)

definitions and associated role structures. Subsequent chapters provide guidance on planning a role engineering effort and on capitalizing on the experiences, positive and negative, of others who have already blazed the trail.

References

[1] Ferraiolo, D. F., D. R. Kuhn, and R. Chandramouli, *Role-Based Access Control*, 2nd ed., Norwood, MA: Artech House, 2007.

[2] Kang Myong, H., et al., "Access Control Mechanisms for Inter-Organizational Workflow," *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, Chantilly, VA, May 3–4, 2001, pp. 66–74.

11

What Others Have Been Doing

The past decade has witnessed significant growth in the development of research topics related to RBAC and in RBAC standards. However, efforts to develop essential domain-specific role content have not keep pace with the development of research results and standards, despite the worldwide interest in RBAC as one form of policy-based access control. RBAC standards and numerous RBAC-related papers from organizations such as the National Institute of Standards and Technologies (NIST), the Association of Computing Machinery (ACM), Health Level 7 (HL7), Organization for the Advancement of Structured Information Standards (OASIS), and the International Organization for Standardization (ISO) are establishing fundamental principals for RBAC implementations. To make these implementations work for an organization, role engineering is needed.

By now, role engineering efforts are going on all the time. Despite the fact that some of the earlier ones turned out to be less than successful, now with experienced firms providing professional role engineering services, more successful efforts have been and continue to be undertaken [1]. In-house efforts have also achieved success [2]. A sampling of these efforts is described on the NIST RBAC Web site http://csrc.ncsl.nist.gov/rbac/RBAC-case-studies.html.

Role Definition Projects

Several projects have been undertaken to define roles for an organization. This implies that these projects sought to define objects, operations, role names, and the combinations of these to define roles. Some role engineering case studies have been published. We recognize that published results do not describe all such efforts and that it is likely that some unflattering details maybe omitted it these public reports. For our purpose these reports are sufficient to document some important characteristics of the role engineering efforts described. In Table 11.1 we summarize these reports using a template that facilitates the comparison of several descriptions [3–5].

Permission Definition Projects

The U.S. Department of Veterans Affairs (VA) is engaged in a large multiyear project to implement comprehensive identity and access management as part of a larger service-oriented security infrastructure [2]. The VA's Veterans Health Administration is adopting RBAC as the means to define healthcare security policies used by this infrastructure. Two needs in this regard were recognized: (1) an enterprise-wide set of roles that would be semantically compatible across a portfolio of applications; and (2) interoperability of access control among VHA and its business partners, which also implies a degree of standardization within the healthcare community. To achieve these needs, a consistent process for defining roles, permissions, constraints, and hierarchies would be needed.

The VHA has for several years now taken the lead in promoting role engineering processes supporting standard role definitions for healthcare organizations. VHA role engineering efforts began more than 5 years ago with the formation of two distinct RBAC task forces, one including several healthcare partners and one internal to VHA. The scenario-based role engineering process resulting from this work was subsequently vetted through both task forces. Based largely upon earlier work by Gustaf Neumann and Mark Strembeck [6], VHA adopted engineering and role definition content models complying with those of the ANSI RBAC standard [7]. This permission definition activity has now been transitioned to the HL7 Technical Committee on Security, with continued input from VHA. As a result, a set of healthcare permissions has been balloted in HL7 and is available for trial use.

Table 11.1
Summary of Several Role Engineering Efforts

Name	Industry Sector	Scope	RBAC Software Used	Types and Numbers of Roles	Time Frame
Veterans Health Administration	Healthcare	Enterprise-wide permissions definition	Generic tools	Permissions only: 108	2 years
Investors bank and trust [3]	Banking	Several systems containing access control data	IBM Tivoli Identity Manager; Eurekify Sage Discovery and Audit	300 business roles and 500 functional roles	6 months
International bank [3]	Banking	Massive cleanup on all major systems; thousands of accounts, groups, and privileges removed	IBM Tivoli Identity Manager; Eurekify Enterprise Role Manager	586 business roles; 935 applications	2.5 months
Insurance company [3]	Insurance	Create an overall database of access rights; identify suspected exceptions such as overlapping, redundant, and out-of-pattern privileges; segregation of duty rules checked against actual privileges	Eurekify Sage Discovery and Audit; Eurekify Sage Business Process Rules	None	100 hours
Large European telecommunications service provider [3]	Telecommunications	48 applications reviewed and analyzed in 4 months; created and verified 80 policies with approximately 15 business process rules per policy	Eurekify Sage Business Process Rules	443 roles	4 months
U.S. utility provider [3]	Energy	Defined the basic enterprise roles and used Sage Reports for auditing and cleaning	IBM Tivoli Identity Manager; Eurekify Sage Discovery and Audit		

Table 11.1 (continued)

Name	Industry Sector	Scope	RBAC Software Used	Types and Numbers of Roles	Time Frame
U.S. health plan provider [3]	Healthcare	Six systems; rule-based roles discovered and deployed in IBM Tivoli Identity Manager as dynamic roles	IBM Tivoli Identity Manager; Eurekify Sage Discovery and Audit	106 roles initially	10 weeks
U.S. leading mortgage lender [3]	Finance	4,000 employees and a growth rate of 200 employees per month; urgent need to deploy RBAC-based identity management	Eurekify Sage Discovery and Audit	160	4 weeks
Major European bank (Dresdner Bank) [4]	Finance	Enterprise-wide level access control services on a variety of systems and applications; applications cannot make access control decisions on their own, but grant access to data on the basis of a centrally provided security profile; more than 60 applications make use of the system; role hierarchies and constraints considered but not implemented	Funktionale Berechtigung (FUB); developed by Dresdner Bank	1,300	Extended
State of Nebraska [5]	Government	Create a single, scalable repository for user identity information to simplify administrative functions and provide for secure identity management; improve data sharing among more than 20 state agencies	Novell Nsure and exteNd solutions		18 months

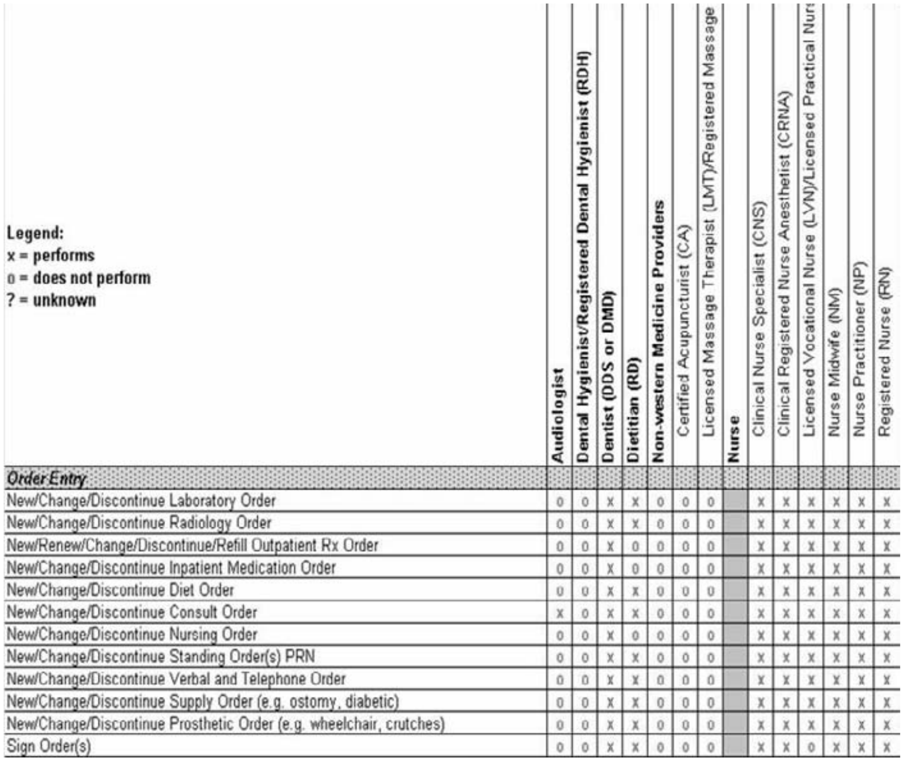
The primary goal throughout the effort has been to standardize permissions and not role names. This is in recognition—gained through VHA, Department of Defense (DoD), and other's experience—that enterprise proprietary role definitions do not suffice for business-business interoperability or secure health information exchange. Given a standard set of permissions, each organization can use these to develop its own role structures. By so doing, even though the roles will be different across organizations, the fact that the roles are composed of standard permissions makes it easier to understand the role definitions from one organization to another. This in turn can provide a degree of interoperability between organizations. Furthermore, at least for healthcare, standard role names such as those published by ASTM [8] can promote actual interoperable roles. These roles would probably be in addition to the organization's internal roles.

VHA distinguished early on between the functional roles described by the ANSI RBAC standard [7] and structural roles as defined in [9]. They extended the definition of functional roles to mean those that are a prerequisite to initiating a session or, within a session, to connect to a resource such as an application.

A national Healthcare RBAC Task Force (TF) comprised of healthcare personnel and healthcare informatics specialists from several organizations was established. The Healthcare RBAC TF had representation from VHA, Indian Health Service (IHS), DoD, and Kaiser Permanente (KP). The Healthcare RBAC TF validated the role engineering process and is promoting the adoption of this process and the standardization of functional role names and functional healthcare permissions within several standards development organizations.

The TF also reviewed and adopted as candidate roles the list of "Healthcare Personnel That Warrant Differing Levels of Access Control" contained within ASTM Standard E1986-98, Standard Guide for Information Access Privileges to Health Information [8]. The standard list of 16 licensed healthcare personnel from the ASTM standard was used to map the healthcare personnel to four high-level and 74 detailed clinical activities performed by licensed healthcare providers. Further, a comprehensive cross-index of the ASTM standard was created for relevant personnel defined by the National Uniform Claims Committee.

During the mapping process, the TF noted that in some cases definitions of healthcare personnel listed in the existing standard were not adequate nor sufficiently consistent for use in the role engineering work. Therefore, the TF presented to the ASTM E31 Committee a set of recommended revisions to the ASTM E-1986 standard. The TF provided 14



Legend:
x = performs
o = does not perform
? = unknown

	Audiologist	**Dental Hygienist/Registered Dental Hygienist (RDH)**	**Dentist (DDS or DMD)**	**Dietitian (RD)**	**Non-western Medicine Providers**	Certified Acupuncturist (CA)	Licensed Massage Therapist (LMT)/Registered Massage	**Nurse**	Clinical Nurse Specialist (CNS)	Clinical Registered Nurse Anesthetist (CRNA)	Licensed Vocational Nurse (LVN)/Licensed Practical Nur	Nurse Midwife (NM)	Nurse Practitioner (NP)	Registered Nurse (RN)
Order Entry														
New/Change/Discontinue Laboratory Order	0	0	X	X	0	0	0		X	X	X	X	X	X
New/Change/Discontinue Radiology Order	0	0	X	X	0	0	0		X	X	X	X	X	X
New/Renew/Change/Discontinue/Refill Outpatient Rx Order	0	0	X	0	0	0	0		X	X	X	X	X	X
New/Change/Discontinue Inpatient Medication Order	0	0	X	0	0	0	0		X	X	X	X	X	X
New/Change/Discontinue Diet Order	0	0	X	X	0	0	0		X	X	X	X	X	X
New/Change/Discontinue Consult Order	X	0	X	X	0	0	0		X	X	X	X	X	X
New/Change/Discontinue Nursing Order	0	0	X	0	0	0	0		X	X	X	X	X	X
New/Change/Discontinue Standing Order(s) PRN	0	0	X	X	0	0	0		X	X	X	X	X	X
New/Change/Discontinue Verbal and Telephone Order	0	0	X	X	0	0	0		X	X	X	X	X	X
New/Change/Discontinue Supply Order (e.g. ostomy, diabetic)	0	0	X	X	0	0	0		X	X	X	X	X	X
New/Change/Discontinue Prosthetic Order (e.g. wheelchair, crutches)	0	0	X	X	0	0	0		X	X	X	X	X	X
Sign Order(s)	0	0	X	X	0	0	0		X	X	0	X	X	X

Figure 11.1 Healthcare Scenario Roadmap.

additional licensed healthcare personnel categories and 34 subcategories within some of those higher-level categories. The ASTM E31 Committee accepted the proposed modifications for future consideration.

Healthcare Scenario Roadmap

Early on in the process, the TF recognized that the effort of identifying work tasks and profiles for all healthcare personnel (licensed, nonlicensed, and noncaregiver) was daunting. To scope the effort, a spreadsheet called the Healthcare Scenario Roadmap was created as a foundational tool to assist in defining the scope of the RBAC modeling effort, as well as to be utilized as a quick reference of healthcare scenarios. The roadmap presents scalable management of user permissions in the form of a list of tasks as a healthcare standard.

Figure 11.1 illustrates a portion of the Healthcare Scenario Roadmap. The vertical axis shows the candidate roles taken from the ASTM set. The abstract permissions are shown on the horizontal axis. The intersecting cells containing an "X" indicate preliminary roles that are generally authorized to perform the corresponding abstract permission, while cells containing an "O" indicate preliminary roles that generally are not authorized to perform the corresponding abstract permission. For instance, the intersection of abstract permission "New/Change/Discontinue Laboratory Order" and preliminary role "Dentist" indicates with an "X" that this role performs this abstract permission. Each of the cells containing an "X" is recorded within a scenario or work pattern and sequence diagram(s).

Healthcare Scenarios

Since scenarios are key to the role engineering process, a source of healthcare-specific scenarios was needed. Two scenario sources were ultimately decided upon: (1) adoption of previous work completed by HL7 Technical Committees' storyboards; and (2) documentation resulting from facilitated sessions of the TF. The healthcare scenarios and discussions were then used within the process to develop a catalog of abstract high-level healthcare permissions which formed the basis for final scenario generation and creating permissions with greater granularity.

Within VHA, the work of deriving permissions is being extended to identifying functional role names with associated permissions. It is expected that other organizations would conduct a similar effort to relate the published HL7 standard permissions to enterprise-specific role names.

In an effort parallel to that of the VHA RBAC TF, the existing VHA VistA healthcare information system was reverse engineered to derive implicitly defined functional roles. The roles so identified have been compared to the permission aggregations of the role engineering effort to achieve a harmonization of the two. This effort will result in a VHA-specific functional role set with standard healthcare permissions.

Task Force Makeup

The VHA RBAC TF was comprised of healthcare personnel, security and technical role engineers. The members were a dedicated team that included physicians, nurses, pharmacists, security architects, developers, and analysts. The TF Chair was a physician with considerable organizational and interpersonal skills along with familiarity in IT. Several other members were also very conversant in federal HIPAA guidelines, federal security requirements, IT, and related policies. In addition, some TF members had experience in other specializations such as healthcare informatics and had participated in standards development organization activities.

The role engineers were from the VHA Health Information Architecture (HIA) Office. Their expertise was in the areas of IT and security. They had hands-on experience with selecting and tailoring the role engineering process. The Functional Analyst Lead of the role engineering effort was both an IT software engineering professional and a Registered Nurse. The role engineers assisted the healthcare domain experts by creating artifacts to contain and display the results of the effort. The Functional Analyst Lead also provided a vital communication function by maintaining a role engineering Web site and producing a monthly newsletter.

Communication Mechanisms

Weekly teleconferences were the primary medium used by the TF for conducting its activities. The agendas for the teleconferences were distributed in advance and minutes were prepared and distributed soon after the teleconference.

The TF held face-to-face meetings approximately every 6 months. The meetings primarily focused on RBAC strategy, the Healthcare Scenario Roadmap, basic roles, and the RBAC architecture.

An RBAC Web site dedicated to the TF's activities was established to contain the VHA RBAC TF artifacts as well as other relevant references such as technology papers and newsletters.

RBAC e-mail groups were used to disseminate information to all parties who expressed an interest in RBAC. There are a few different RBAC e-mail groups, includingthe Healthcare RBAC TF, the VHA RBAC TF, the VHA business partners, and outside interested parties. E-mails were sent on a regular basis regarding the latest RBAC happenings.

A newsletter was developed and distributed to all interested RBAC parties on a monthly basis. The newsletter contained pertinent Standards Development Organization (SDO) information and synopses of the latest RBAC research papers and standards information. The newsletter always contained a hyperlink to the RBAC Web site.

Exit Criteria

Exit criteria were defined and used as guideposts to indicate when the TF goals were met in a satisfactory manner. Since this was a collaborative effort, the collaboration was needed to collectively evaluate the adequacy of satisfying the defined exit criteria. All TF deliverables were evaluated with the exit criteria in Table 11.2.

Work Method of the Task Force

Scenario Identification

Two sources of healthcare scenarios were used: HL7 storyboards and facilitated sessions with the TF members. The storyboards, provided courtesy of HL7, consisted of fictitious settings representing typical healthcare encounters and related IT functions. The storyboards were not ideal as healthcare scenarios for three reasons: (1) they represented the activities of multiple actors, which is contrary to the role engineering process; (2) they did not provide adequate coverage of healthcare activities; and (3) historically, HL7 has been a messaging standard, so use of these storyboards was not an easy fit. However, as used to identify permissions and not roles, they were adequate

Table 11.2
Exit Criteria for Work Products

Completeness	Did the goal or work product cover all recognized topics?
	Were these topics covered adequately?
Correctness	Were all parts of the goal or work product free from significant errors?
Internal consistency	Were all parts of the goal or work product consistent with each other?
External consistency	Were all parts of the goal or work product consistent with other accepted sources?
Generality	Was the goal or work product free of ad hoc assumptions and locally defined components?
Simplicity	Were the parts of the goal or work product free of complex language or analysis that may impede the document's use?

to "prime the pump" for the effort. As a result, the TF was required to fill in the gaps and create scenarios to complete the coverage of the healthcare domain.

Facilitated Sessions

A series of weekly teleconferences provided the primary means for the TF members to provide input and to discuss the labeling and categorization of permissions. Dedicated staff provided pre- and postmeeting support to document, organize, and publicize the ongoing results. The staff maintained a list of action items to ensure that all issues were appropriately addressed at each meeting. In addition to the weekly teleconferences, an annual face-to-face meeting was held to provide more opportunity for team interaction and resolution of issues.

Outreach Within and External to the Organization

Dedicated staff, and others enlisted for the purpose, publicized the TF activities and contacted additional subject matter experts as well as systems developers in the VHA organization to solicit material on permissions and roles. Each system development project was contacted and provided forms for input of RBAC-related definitions. A periodic TF newsletter was prepared and distributed to VHA as a whole and to external organizations with interest in role engineering.

Existing and Emerging Standards

Several standards organizations have been defining standards for RBAC. Some of these are concerned with RBAC itself and some have relevance for role engineering.

Health Level 7

HL7 is an international organization that defines standard messages and other constructs to facilitate the syntactic and semantic interoperability of healthcare systems. To assist with the interoperability of RBAC implementations, HL7 has undertaken an effort to standardize a set of healthcare permissions. The prior VHA work in this area was used as a point of departure for the HL7 work. A draft standard for trial use has been published and the work continues.

RBAC Standard

The RBAC standard [7] consists of two major components: a role model and a set of functional descriptions. The significance of this standard for role engineering is that in our role engineering work we continually refer to the definitions from the role model in the RBAC standard. Thus, the items we define in a role engineering effort—User, Role Name, Permission (Operation, Object), Constraint, and Hierarchy—are as defined in the RBAC standard.

RBAC Implementation Standard (Interoperability of Role Definitions)

At present, an effort is underway to define an RBAC implementation standard [10]. The two purposes of this standard are to define what it means for a system to be compliant with the RBAC standard and also to define a means for two RBAC systems to exchange data on role and user definitions for purposes of interoperability. The compliance part is intended to assist in the process of comparing two RBAC systems on a common basis, while the interoperability part is intended to provide definitions on how two RBAC systems can exchange metadata on their role structures.

ASTM Role Names and Privilege Management Infrastructure

As described earlier, the ASTM Committee for Healthcare Informatics has developed a standard that contains a set of role names for healthcare [8]. The significance of these for role engineering is that the role names may be used as candidate roles with which to develop healthcare permissions. A secondary possible use for the standard role names would be as structural roles that contain permissions that permit a user to connect to a network, application, server, or other resource. Using the standard role names can aid communication between different parts of the organization, or with other organizations.

Role Engineering Standard (HL7, Possibly INCITS)

HL7 has adopted the scenario-driven role engineering process [6] as tailored by VHA [2] as a draft standard for trial use. This is a simplified version of the process defined by Neumann and Strembeck that omits such items as hierarchies and some constraints.

In addition to this available draft standard is a proposal for a different role engineering standard that would include more that one approach with the objective of producing results that conform to a set of norms. Thus, this possible new standard relates directly to role engineering specifically as a standard process for gathering requirements for roles and for composing this data into a role structure with possible hierarchies and constraints. This standard, as is the draft RBAC implementation standard mentioned earlier, would be based on the RBAC standard.

OASIS XACML RBAC Profile

The OASIS XACML standard provides a set of constructs in the XML language for the implementation of access control policy, including policy decision points and policy enforcement points. The XACML RBAC profile [11] is constructed within XACML to define some of the components of the RBAC model in the RBAC standard. These include roles, permissions, users, and constraints.

The significance of the XACML RBAC profile for role engineering is that one means of representing permission definitions may be to use the RBAC profile and XACML itself.

Table 11.3 identifies the major RBAC standards and provides some background information on them.

RBAC Research Activities

Considerable research into RBAC topics has been conducted over the past decade (see, for example, http://www.acm.org/sigs/sigsac/sacmat.html). We

Table 11.3
Major RBAC Standards

Standard	**Description**	**Status**	**Relevance to Role Engineering**
ANSI INCITS 359-2004 American National Standard for Information Technolgy—Role-Based Access Control [7]	U.S. RBAC Standard—contains an RBAC reference model and an RBAC system and administrative functional specification	In force	RBAC reference model provides concepts and terminology for role definitions
ANSI INCITS Role-Based Access Control Implementation Standard [10]	U.S. RBAC Implementation Standard—interprets ANSI INCITS 359-2004 for compliance and interoperability	Draft—not balloted	RBAC reference model provides concepts and terminology for role definitions
ASTM Standard Guide for Information Access Privileges to Health Information [8]	List of healthcare-related role names	In force	Provides healthcare candidate role names for possible use in defining healthcare permissions
OASIS XACML Profile for Role-Based Access Control (RBAC) [11]	Industry standard—provides XML language constructs to implement RBAC features of core, hierarchical, and dynamic separation of duty	In force	Once roles have been defined, XACML may be used to represent the resulting policy constructs
Health Level 7 (HL7) RBAC Role Engineering Process [2]	Based on [6]	Draft standard for trial use	Provides guidance on defining core RBAC roles
HL7 RBAC Healthcare Permission Catalog [2]	Standard permissions for healthcare	Draft standard for trial use	Provides a standards set of healthcare permissions
HL7 Healthcare Scenario Roadmap [2]	Standard permissions mapped to ASTM candidate roles	Draft standard for trial use	Illustrates a working set of permissions definitions and related candidate roles
HL7 RBAC Healthcare Scenarios http://www.hl7.org	Healthcare scenarios for use in defining healthcare permissions	Draft standard for trial use	Illustrate scenarios that may be used to identify roles

highlight some of the research topics here as being relevant to the role engineering process discussed in this book.

Context-Sensitive Permissions

In [12], Strembeck and Neumann follow up their previous work on scenario-driven role engineering [6] with a model for a type of constraint on permissions that captures contextual data. This constraint, called a context constraint, consists of predefined conditions that are tested in real time as the permissions are checked in making an access control decision. An example of one of these context conditions would be the current value of a data item in a database. The concept of a context constraint can be a powerful one, in that an access decision could depend on the source or other characteristic of a unit of data. Thus, a user's ability to access a class of data could be automatically restricted by assigning values to data objects. As these values were dynamically changed, so would the users' effective permissions change accordingly.

The significance of this topic to role engineering is that the context constraints would have to be defined. These definitions would consist of sets of context conditions with their specific context definitions and expected values.

Automatic Assignment of Roles to Users

This topic may point to the future of how RBAC is implemented. As usually envisioned, a user would manually activate a set of his or her assigned roles to perform a particular activity. This activation and deactivation of roles by a user can be cumbersome.

Two versions of this have been called rule-based RBAC and rule-based provisioning of RBAC. Their characteristics are summarized in Table 11.4 [13].

Thus, rule-based RBAC provides a highly dynamic environment in which roles are automatically assigned and designed to users, while rule-based provisioning of RBAC also provides automated assignment of user to roles but not at run time. Each approach has its pros and cons. These can be overcome to an extent through system design.

A variation on these two approaches to automatic assignment of roles to users is an idea of one of the authors of this book. This approach might be called "Defining Roles for a Fluid and Dynamic Organization." It can be

Table 11.4
Rule-Based RBAC and Rule-Based Provisioning of RBAC

Rule-Based RBAC	**Rule-Based Provisioning of RBAC**
Roles are not assigned to users manually. Instead, users are assigned to roles dynamically at run time by the system's using user attributes to determine proper role assignments. These attributes could include organization, job function, or location.	A more static version of rule-based RBAC, this approach brings users, their attributes, and further access right information into the RBAC system. The process is normally automated by taking information from HR databases, corporate directories, or other information bases in the enterprise. Such databases contain information about employees entering or leaving the company and data such as employee number, organizational unit, location, or job description.
Advantages of This Approach Are:	**Advantages of This Approach Are:**
The approach offers most of the advantages of RBAC, including the possibility to use role hierarchies. Rules feature dynamic role assignments based on user attributes and thus reduce administration effort. If changes in the role assignments are necessary (e.g., due to organizational or operational changes), no manual assignments and deassignments of roles are necessary.	As the RBAC system contains explicit role assignments, the administrator has a good overview of the actual authorizations that a user has in the system. For the same reason, the system provides good auditing capabilities, which is very important in many industries such as banks and insurance companies. Rules are used to compute the role assignments by exploiting information from existing databases. Using these rules, a high grade of automation can be achieved, thereby considerably reducing the manual administration effort. Automation using rules also reduces the probability of errors and leads to a higher security level as it prevents errors made in manual administration.
Disadvantages of This Approach Are:	**Disadvantages of This Approach Are:**
Large organizations probably have a large number of rules, making it difficult to maintain an overview of who has which permissions. The missing overview makes it especially difficult to fulfill auditing requirements. It is often difficult to foresee the impact of a new rule or the modification of an existing rule. There exists a strong possibility of conflicts or inconsistencies among rules. The chances of this increase rapidly as more rules are added, and at some point conflicts become almost inevitable. Incompleteness is another concern—that is, are all possibilities covered?	It is often difficult to foresee the impact of a new rule or the modification of an existing rule. Changes in rules or attributes do not take effect immediately on permissions of users. Security becomes dependent on user attributes assigned by and stored in a human resources repository, or somewhere else. Security is less direct, and thus may be harder to assure since more components are involved. It may be possible to manipulate employee attributes to obtain permissions that should not be assigned. Responsibility for security may become distributed among multiple departments within the organization, which may lead to both technical and interpersonal challenges.

Source: [13].

compared in concept to a heads-up cockpit display in a military aircraft, where the user's focus automatically changes as the system tracks environmental factors such as speed, position, and distance to target.

This approach contains the following features:

- Job functions for an individual can be multiple in parallel or in series.
- Role definitions would be in terms of clusters of permissions.
- Cluster formation would be dynamic and follow rules of authorization of a finite number of roles. The user would not select the roles to be used; rather, they would be invoked automatically according to the user's attempts to accomplish some task.
- Role definitions would change automatically behind the scenes.
- Auditing would be performed not on role usage but on permission usage. The roles corresponding to these permissions would be computable and could be used in auditing if desired. These roles would be considered canonical and it is these that would be assigned to users, either manually or automatically.

The significance for the role engineering process of automatic role assignment to users lies in the fact that in addition to defining the permissions, the process should also include definition of the rules and data values that will control the role assignment.

Multihierarchy Role Relationships

This topic recognizes that there may be more that one valid and useful hierarchy of roles in an enterprise. This possibility is discussed in [14]. As Figure 11.2 illustrates, these types of hierarchy could include authority structure, organizational structure, or responsibility structure. When these various types of hierarchy are determined to be needed, the role engineering process must take their definition into account. Data on them must be elicited from the role engineering subject matter experts in a top-down approach or derived form existing data sources in a bottom-up approach.

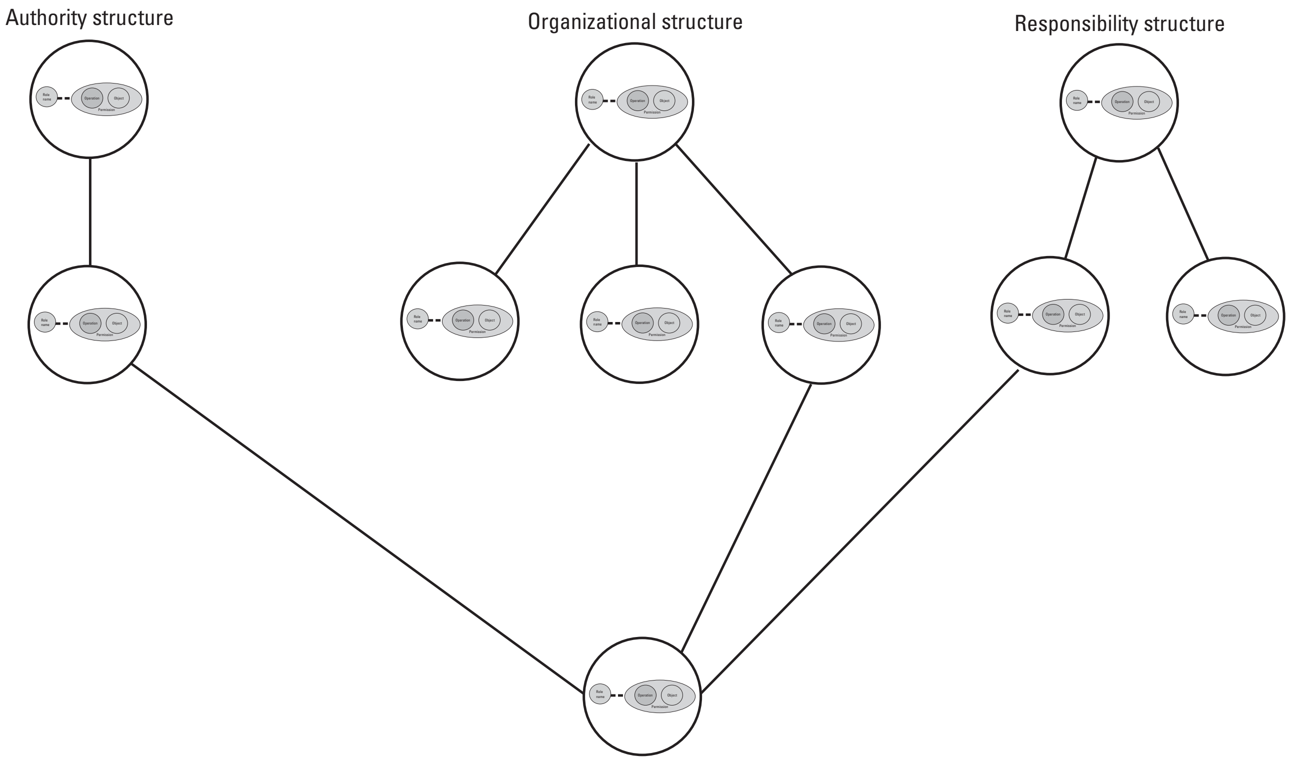


Figure 11.2 Multihierarchy role relationships. (*Source:* [14].)

Economic Analysis of RBAC

The staff of NIST have conducted research into various facets of economic analysis of RBAC [15, 16]. Another source of guidance on economic analysis is the work done by Capers Jones over the years. While Jones' latest book [17] does not address role engineering economics directly, it provides approaches and processes for estimation of software costs that can be adapted to role engineering. Future research could exploit for role engineering these more traditional estimation methods originally developed for software engineering.

Dynamic Role Definitions

The U.S. Navy Enterprise Dynamic Access Control (EDAC) [18] allows dynamic assignment of roles and permissions depending on user and environment characteristics. For example, as user information changes in human resources repositories, the updates are automatically reflected in role assignments. This system may fulfill the access control needs of highly dynamic organizations. This interesting system will require very careful role design and testing to ensure that even in a highly dynamic environment the access control policy is continually enforced.

Testing and Assurance of RBAC Policy Definitions

Since the result of role definitions is intended to accurately reflect the access control policy, testing of roles and their structures is critical. Relevant research results are being produced that can assist in approaching the role testing process. In particular, research on testing of access control policies is presented in [19–21].

SACMAT and ACSAC

Two annual conferences are valuable sources of forward-looking information and analysis on role engineering. These are the ACM Symposium on Access Control Models and Technologies (SACMAT) and the Annual Computer Security Applications Conference (ACSAC). This is exhibited by many of the references listed in this book. Until 2000, the SACMAT symposium was called the ACM Workshop on Role-Based Access Control. Thus, it has a strong focus on matters relevant to role engineering.

References

[1] Eurekify, Inc., *Role Management Is Fast Becoming a Required Aspect of Identity Management*, February 2006.

[2] VHA-RBAC, http://www.va.gov/RBAC/documents.asp.

[3] Eurekify, http://eurekify.com/customers.case.studies.asp.

[4] Schaad, A., J. Moffett, and J. Jacob, "The Role-Based Access Control System of a European Bank: A Case Study and Discussion," *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, Chantilly, VA, 2001, pp. 3–9.

[5] Novell, http://www.novell.com/identity/access/success.html.

[6] Neumann, G., and M. Strembeck, "A Scenario-Driven Role Engineering Process for Functional RBAC Roles," *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, Monterey, CA, June 3–4, 2002, pp. 33–42.

[7] InterNational Committee for Information Technology Standards (INCITS), ANSI INCITS 359-2004 American National Standard for Information Technology—*Role Based Access Control*, February 3, 2004.

[8] *ASTM E 1986–98 Standard Guide for Information Access Privileges to Health Information,* approved October 10, 1998, November 1998.

[9] International Standards Organization, Health Informatics—Functional and Structural Roles, Draft Standard for Comments, TC 215/WG4/N214, ISO/PDTS 21298, January 15, 2004.

[10] InterNational Committee for Information Technology Standards (INCITS), Proposed American National Standard for Information Technology—*Role Based Access Control Implementation Standard*, Draft, 2007.

[11] OASIS, *XACML Profile for Role Based Access Control (RBAC)*, Committee Draft 01, February 13, 2004, http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf.

[12] Neumann, G., and M. Strembeck, "An Approach to Engineer and Enforce Context Constraints in RBAC Environments," *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003)*, Como, Italy, June 2–3, 2003, pp. 65–79.

[13] Kern, A., and C. Walhorn, "Rule Support for Role-Based Access Control," *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT 2005),* Stockholm, Sweden, June 1–3, 2005, pp. 130–138.

[14] Crook, R., D. Ince, and B. Nuseibeh, "On Modelling Access Policies: Relating Roles to their Organisational Context," *Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering* (RE'05), Paris, France, August 29–September 2, 2005.

[15] Ferraiolo, D. F., and J. F. Barkley, "Comparing Administrative Cost for Hierarchical and Non-Hierarchical Role Representations," *2nd ACM Workshop on Role-Based Access Control*, Fairfax, VA, November 6–7, 1997.

[16] Gallaher, M. P., A. C. O'Connor, and B. Kropp, *The Economic Impact of Role-Based Access Control*, Planning Report 02-1, National Institute of Standards and Technology, 2002.

[17] Jones, C., *Estimating Software Costs: Bringing Realism to Estimating*, 2nd ed., New York: McGraw-Hill, 2007.

[18] NIST-RBAC/EDAC, http://csrc.nist.gov/rbac/edac.html.

[19] Hu, V. C., et al., "Conformance Checking of Access Control Policies Specified in XACML," *Proceedings of the 1st IEEE International Workshop on Security in Software Engineering (IWSSE 2007)*, Beijing, China, July 2007.

[20] Martin, E., T. Xie, and T. Yu, "Defining and Measuring Policy Coverage in Testing Access Control Policies," *Proceedings of the 8th International Conference on Information and Communications Security (ICICS 2006)*, Raleigh, NC, December 2006, pp. 139–158.

[21] Martin, E., and T. Xie, "A Fault Model and Mutation Testing of Access Control Policies," *Proceedings of the 16th International Conference on World Wide Web (WWW 2007),* Banff, Alberta, Canada, May 2007, pp. 667–676.

12

Planning a Role Engineering Effort

The Importance of Good Planning

Planning is a combination of forecasting and controlling the future, as noted by Cleland and Kocaoglu [1]. Each of these aspects complements the other: forecasting without using the results to control the future is of little use, and attempting to control the future without the benefit of forecasting has an even weaker basis. With this in mind we note that planning for role engineering should address both forecasting of how an effort should be structured and executed as well as establishing goals, adopting strategies and methods, accomplishing staffing, and setting up control mechanisms.

Forecasting entails the gathering of data on the performance of previous projects. For this reason, each role engineering effort should contain a component whereby data is gathered on the adequacy or inadequacy of the scoping, the effectiveness or lack of effectiveness of strategies and methods, the adequacy or inadequacy of staffing, and the effectiveness of the controlling mechanisms. Gathering and reporting on this data will serve to support forecasting for possible future efforts and also provide visibility into the workings of the current effort for enterprise management. In turn, this visibility for management can facilitate the justification of future role engineering efforts.

With effective forecasting, the scope, strategy, methods, and staffing for the role engineering effort can be established on a firm and defensible basis. Providing that these areas are adequately addressed, the control aspect of planning becomes the central portion of the planning process. Because it is impossible to be infallible in predicting the future and in completely controlling the future, it will be necessary to make mid-course corrections during the conduct of a project. Assisting in the correction process are the availability of current performance data as well as a continual assessment process and appropriate feedback to the team.

Preparing a good plan and following the plan can help to minimize uncertainty in the project and to communicate essential information to all concerned with the project. On the other hand, failure to plan a role engineering effort adequately, as with other types of projects, can lead to disaster. Without a good plan that is followed, chaos can ensue and the effort can quickly deteriorate into failure. This failure can be catastrophic or merely an unfortunate occurrence. Neither of these results is satisfactory and they can be avoided.

One of the common reasons for project failure is a lack of definition of objectives [2]. The objectives should be specific. For a role engineering effort they may include the following for each of the domains to be addressed:

- A set of candidate roles will be defined.
- Constraints will be identified.
- A permissions catalog for each included domain will be prepared.
- A set of enterprise roles will be defined.
- The permissions will be assigned to the enterprise roles.
- Constraints (permission-permission, permission-role, and role-role) will be applied to the role definitions.

A second common reason for project failure is inadequate communication [2, 3]. This refers to both intra-team communication and extra-team communication. It is essential for each team member to be aware of the activities of the team as a whole and for management and other stakeholders to be aware of the progress and results of the team. One overall aid to good communication is to use consistent terminology, preferably from appropriate standards, for example the RBAC standard or the XACML RBAC profile [4, 5]. This will be especially helpful when dealing with outside organizations or contractors if they are familiar with the standard terminology.

This communication can avoid costly misunderstandings about content and about project and program management concerns.

Justifying the Project

Prior to beginning a role engineering effort, a proposal for conducting the effort must be justified and approved. The proposed effort must then be qualified for acceptance by management. In an environment of competition for scarce resources among potential projects, the proposed effort must be sufficiently described and its planned return on investment for the enterprise must be presented. There will be various types of risk associated with the effort, and these must be identified and addressed to complete the justification process. The proposed role engineering effort will likely be weighed against other proposed efforts for anticipated return on investment, meeting corporate requirements, feasibility, risk, and cost. When this is the case, it will be necessary to identify and quantify these factors to permit a valid evaluation of alternatives to be conducted by technical staff and, ultimately, for decisions to be made by management.

The justification for the project will typically include estimates of staff and funding. Chapter 13 addresses staffing considerations. Staffing estimates must be verified and both the staffing and funding estimates must be expanded into executable plans for conducting the effort. Staffing will include both subject matter experts and professional support staff. We recommend that these two categories be kept separate. Subject matter experts should concentrate on exercising their professional expertise and should not spend time on IT issues to support the role engineering work. Professional support staff should refrain from inserting their opinions into the substantive discussions of the team. This division of labor will strengthen the team as a whole and maximize its value for the definition of roles.

Planning the Project

Two basic approaches may be taken for conducting the proposed role engineering effort. One approach is to conduct an independent role engineering effort at the enterprise level. The other approach is to include role engineering within a system implementation effort. It would be unlikely for a single implementation effort to result in a complete definition of roles for the enterprise, so it will to be necessary to perform role engineering on several system implementation projects before a complete set of enterprise roles is defined. The role engineering results from each of these individual implementation projects will have to reflect a common set of goals and expectations, and the results from the projects will need to be coordinated. For these reasons it would be simpler to conduct an independent enterprise-level effort than to coordinate the role definition results of multiple implementation efforts. Of course, it is also possible to combine an enterprise-level effort with any number of implementation-oriented efforts.

Once it has been decided that a role engineering effort will be initiated and supported, planning must begin. Some preplanning should be undertaken to address the areas of ongoing performance evaluation and cost tracking that should take place during the pursuit of the role engineering effort. This will include the establishment or adaptation of a metrics program to measure and record progress, effectiveness, and cost.

It is advisable to draw up a charter for the group who will be carrying out the effort and to have that charter approved by management. Because of the need for staff, funds, and time, this commitment is very important to the success of the effort. Also, since subject matter experts' time will be required, management commitment and also individual commitment must be obtained in advance of the project start.

Staffing is considered in Chapter 13. Once a staffing plan has been developed, implementation of the plan will require the identification and selection of available individuals to carry out the effort. This staffing process is crucial to the success of the effort. The quality and appropriateness of the staff can determine success in meeting the time and resource goals and can also determine the quality and usefulness of the results.

A project plan will include more than a schedule and milestones. While these are important, also important are identification of resources by type and level of effort, expectations about results both along the way and at the conclusion of the project, and a communications plan. The expectations can be represented by narrative descriptions of milestones that apply to each phase of the project. As these milestones are reached, the team can review these descriptions of expectations and discuss how well they were met and how they might be better met during the future conduct of the project.

Communications Plan

A communications plan should be part of the project plan. It will identify how the project team members will interact with one another to interchange data, review work items, and to discuss topics and action items. Use of a collaboration tool should be considered as part of the planning process. Either a Web-based or a workstation-based tool could be employed, with pros and cons to be considered for each type of collaboration tool. Prior to initiating the project, all members requiring access to the tools should be set up and ready to go. If it is decided that a collaboration tool will not be used, the alternative methods of transferring data among team members should be described in the communications plan. Other modes of communication such as face-to-face meetings and conference calls should also be defined in the plan.

The Planning Process

Developing a plan for a role engineering effort can be approached by answering a set of general questions. The answers obtained will form the foundation of information for the plan. The questions to be addressed for planning are: Why? What? Who? When? How? and Where? These questions cover the areas that need to be addressed by a comprehensive plan. In Table 12.1 we define these one at a time for our purposes.

Table 12.1
Questions to Be Addressed for Planning a Role Engineering Effort

Question	Definition	Examples
Why?	The purpose of the effort	Create a role structure that reflects the workflows and security policy of the enterprise
What?	The components of the plan	Gathering data, defining components of roles, assembling components into roles, verification and validation; business domains (financial, mission oriented, management)
Who?	The people carrying out the plan	Role engineering specialists, subject matter experts
When?	The relative or absolute time frame	For role engineering as part of a system implementation project, the phases of the SDLC; for an enterprise-level independent role engineering effort, as long as it takes to define the roles identified in the scope of the effort
How?	The components to support the plan	Resources, role engineering processes, best practices, tools
Where?	The various considerations regarding the options for conducting the effort and to a limited extent the scope of the effort	In-house, out of house, parts of the enterprise

Discussion of the Six Questions

Why? The question of Why? should have been answered by the time a role engineering effort is being planned. However, it is a good idea to revisit the reason for wanting to conduct such an effort and to make this reason known to management, subject matter experts, technical staff, and additional stakeholders.

What? One of the first decisions that must be made is in regard to the scope of the intended effort. Is the role engineering effort intended to establish permissions, role names, role names with permissions, roles with constraints, roles with hierarchies? What domains are to be addressed: mission oriented, administrative, all? What IT systems and resources are to be addressed? These are some of the questions that must be answered concerning scope.

Who? Staffing the effort is the next consideration after determining a process to be followed. This topic is treated in detail in Chapter 13. As mentioned, the availability of staff will directly affect the scope and possibly the process used. A commitment of role engineering staff for the estimated duration of the effort (see below) is a key factor that will determine the success of the effort. Thus, it is recommended to postpone or even forego the role engineering effort if adequate staffing is not available to see the effort through to a productive conclusion.

When? A portion of this question was addressed in Chapter 3. Another portion of the question deals with which systems will be addressed first where the scope includes developing structural or functional system roles.

How? Once the intended scope has been defined, it will be necessary to establish a role engineering process for use. The material in this book can help with that. A top-down approach versus a bottom-up approach needs to be considered. A full-blown scenario-based top-down approach versus a streamlined actor-to-permission approach are options to be considered. If a bottom-up approach is decided upon, it will be necessary to identify the systems or other sources of the role and permission data. It is likely that any available role engineering process will require tailoring to meet local requirements. However, as this is done, the underlying principles of the process should be borne in mind so that the best possible role definitions are obtained with the optimal amount of effort. Once that amount of effort has been estimated, it may be desirable to revisit the decisions made earlier to achieve a better fit between available resources and planned outcome.

Where? This question, rather than referring to location, is more closely related to the questions of What? and Who?. The determination as to whether the resources will be in-house, out of house, or a combination will depend on the experience level of in-house personnel and their availability for a role engineering effort. The determination of which parts of the enterprise will be included in the role definitions will depend on enterprise priorities, availability of funds, and, perhaps, the significance of legacy systems.

Level of Effort

While it is not possible to predict precisely the level of effort that will be required for each role engineering project, some basic data points are available that can assist in resource planning. In the VHA RBAC Task Force effort where about 65 high-level permissions were defined, one role engineering analyst and from two to five subject matter experts met weekly for 2 years. SMEs spend 3 hours per week on the role engineering effort. This amounts to approximately 576 hours per year: 192 total for SME and analyst time on 48 weekly calls and another 384 hours additional analyst time at 8 hours per week. These parameters are summarized and expanded in Table 12.2. The expansion uses the data points as a basis for estimating variations in staffing and time frames. A linear extrapolation was used. Practice may indicate a different function for the expansion. The figures are provided as a possible starting point for planning.

Table 12.2
Projected Level of Effort

Number of Permissions (High-Level)	Role Engineering Analyst	Subject Matter Expert*	Yearly Meetings	SME Time	Analyst Time	Yearly Total Time
(Estimated) 25	1	1–2	18	55	18 + 147 = 165	220
(VHA Experience) 65	1	2–5	48	144	48 + 384 = 432	576
(Estimated) 150	2	5–12	111	332	111 + 886 = 997	1,329

*Dependent domains to be addressed.

Key Milestones

A logical next step in the planning process is to establish key milestones for the effort. The milestones can include such items as identification of actor names to be used for candidate (scaffold) roles, identification of any scenarios that may be available, creation of scenarios where these are not available, extraction of a certain number of high-level permissions from the scenarios, association of a certain number of actors with high-level permissions, normalization of the set of high-level permissions, and creation of a permissions catalog.

Measuring Progress

Since the amount of effort to be expended for the effort should be proportional to the value derived from the effort, it is necessary to measure progress during the effort and to assess the degree of quality results being obtained over time. If it is determined that progress is not sufficient given the amount of effort being expended, it will be advisable to reassess the parameters of the effort and to make any necessary course corrections to bring progress into line with expended effort. One way that was used by the VHA to maximize efficiency in the use of SME time was to have the analyst do the majority of the work that did not require SME participation. Data manipulation and analysis work as well as preparation for the weekly teleconferences were performed apart from the SMEs such that the SMEs could concentrate their time on identifying and validating high-level permissions. There tended to be a temptation on the part of the SMEs to engage in analysis and data-related discussions, so it was decided that the analyst would work with the data and facilitate the meetings to make the optimal use of the SMEs' time.

It is important to measure the progress of the effort to justify the expenditures of funds and other resources and to ensure a successful completion. If progress is not being made according to plan, there should be a course correction made that will bring the effort back into alignment with its goals. Thus, the goals for the effort must be established before the effort starts. With the goals stated, it then must be possible to measure progress against the goals.

Table 12.3 presents some categories that may be used to track progress. Depending on the scope of the effort, some or all of these may be relevant.

Table 12.3
Progress Tracking Parameters

Goal	**Description**	**Unit of Measure**	**Typical Value or Range**
Permissions defined	High-level permissions defined	Number per month	6–12
Permissions realized	High-level permissions realized as lower level permissions	Number per month	2
Role names defined	Top-level role names before possible decomposition into junior roles	Number per month	8
Constraints defined	Permission-permission, role-role, and user-role constraints	Number per month	2
Hierarchies defined	Hierarchies created from single roles	Number per month	1
Roles reviewed	Review by independent analyst	Number per month	4–6
Roles tested	Test for enforcement of access control policy	Number per month	2–4
Roles accepted	Acceptance for inclusion in repository	Number per month	2
Roles documented	Documentation and publication of role definitions	Number per month	2

Numerical parameters are to be set for each effort; however, some typical parameters are provided as guidance.

Results of the effort should be assessed on a continual basis, with monthly reporting. A schedule for completion of the various goals should be established at the outset. The established values for measuring progress should be used to compute the extent that the effort is behind or ahead of schedule. It is important not to change the benchmark values during the course of the effort, for example restarting the evaluation process several times with the same or different values. If the progress measurement shows a negative condition (i.e., that the goals are not being met), a course change should be made to improve the rate of progress, but the measurement of progress should remain fixed.

Additional Tracking

In addition to tracking progress, it is advisable also to track any difficulties that are encountered. Perhaps some of these difficulties will be due to a lack of specialized tools. Where this is the case, a parallel effort may be undertaken during the role engineering effort, or subsequent to it, it to identify and possibly acquire tools that would alleviate some of the difficulties encountered. Role engineering tools and their acquisition are discussed in Chapter 9.

Tracking costs will customarily be done on a system development project. Costs should also be tracked for an independent enterprise-level role engineering effort. In addition to the usual tracking of cost per unit time or per work package, for role engineering it is advisable to track costs per unit of role definition results as well. When costs are accurately tracked against role definition results, it becomes feasible to compute return on investment figures. This data can prove useful in assessing the worth of the project to the organization and for making future trade-offs when trying to evaluate proposed role engineering projects against other proposed projects.

Summarizing the Plan

The contents of the plan ideally will consist of the items discussed, as listed in Table 12.4.

Summary

We have tried to emphasize the importance and value of planning any role engineering effort. The prediction value of the planning process alone, as well as the resulting plans, can justify the time and effort expended to develop a comprehensive plan. When an effort is made to ask the relevant questions and adequate answers to these questions are obtained, the result will be a far better understanding of what is intended to take place during the project and how this will be accomplished. The additional effort to organize the project for efficiency and effectiveness will complete the planning process and serve to promote its success.

Table 12.4
Contents of the Plan

Plan Component	Description
Justification for the project	A summary of the successful project justification
Communications plan	How team members will collaborate and exchange results
Purpose	Motivation for the effort and what needs to be accomplished
Components	The activities covered by the plan
Staffing	The types and numbers of people to carry out the plan
Time frames	Project phases and durations
Support resources	What is needed to carry out the plan
Strategy and scope	Approaches to carrying out the plan
Level of effort	Amount of work to be expended
Key milestones	Decision points and deliverable dates
Metrics	Measuring success factors and gathering lessons learned data
Additional tracking	Identifying difficulties to be overcome by technology or other means; recording costs for use in measuring success of the present effort and predicting resources for future efforts

References

[1] Cleland, D. I., and D. F. Kocaoglu, *Engineering Management*, New York: McGraw-Hill, 1981.

[2] Cleland, D. I., and L. R. Ireland, *Project Management*, New York: McGraw-Hill, 2002.

[3] Cleland, D. I., *Project Management*, Blue Ridge Summit, PA: Tab Professional and Reference Books, 1990.

[4] InterNational Committee for Information Technology Standards (INCITS), *ANSI INCITS 359-2004 American National Standard for Information Technology—Role Based Access Control,* February 3, 2004.

[5] OASIS, XACML Profile for Role Based Access Control (RBAC), Committee Draft 01, February 13, 2004, http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf.

13

Staffing for Role Engineering

A successful role engineering effort depends on the availability of a team of qualified individuals with effective leadership of the team's efforts. Without such a qualified team, it will not be possible to achieve success. For a role engineering effort that is to be totally or primarily an in-house effort, the importance of staffing will be paramount. The importance of staffing will be less so if the effort is outsourced partially or totally.

In our discussion of staffing for role engineering we focus on an in-house, top-down process effort as a model (Option A in Table 13.1). We recall that a top-down process is one where roles are defined by a requirements gathering and engineering process using subject matter experts (SMEs). A bottom-up process is one where existing systems or system components are examined, possibly using automated tools, to discover implicitly defined roles and to make these roles explicit and mutually consistent. As discussed in Chapter 6, before a decision to use a bottom-up approach is made, an assessment of the available material in existing systems should be conducted. Only where the material is expected to provide quality role information should resources be expended for any bottom-up work.

We use this model to discuss staffing issues and recommendations because the importance of staffing is strongest for this model. Variations on the model include situations where the effort is outsourced, where a bottom-up approach is included, and where a combination of in-house and outsourced staff are employed. These variations are addressed in addition to

Table 13.1
Staffing and Process Options

Option	In-House/Outsource	Process Type
A	In-house	Top-down
B	Outsourced	Top-down
C	Outsourced	Bottom-up
D	Both	Both

the in-house, top-down model. The basic model and the other options for staffing and process approach are shown in Table 13.1.

Actual staffing for role engineering pertains primarily to the role engineers and professional support personnel that will be needed. Much of the work will also be done by the subject matter experts, and these individuals are essential, but they are not considered to be staff as far as this effort is concerned. To some extent, the SMEs are interchangeable, while the dedicated staff should be relatively permanent. This chapter focuses on staffing for the role engineering and professional support functions.

Two primary considerations regarding the staffing of a role engineering effort are cost and effectiveness. Added to these would be risk and stability. The cost consideration can be part of the justification process. Effectiveness is directly related to the benefits to be realized from the role engineering effort. Risk relates to the likelihood of a successful effort, and stability relates to the continued composition of the role engineering team over time. Risk and stability depend on several factors, but they all must be addressed to ensure success.

Effectiveness Considerations

Option A For an in-house, top-down effort, effectiveness comes into play with the required sufficiency and appropriateness of the selected staff members. Of course, the team leader must be experienced and otherwise well qualified. Effectiveness can be enhanced by use of appropriate analysis and collaboration tools.

Option B For an outsourced, top-down effort, effectiveness comes into play with the selection of a firm that will provide sufficient and appropriate staff members and an effective team leader. Whether or not analysis and collaboration tools are used will be decided upon by the firm providing the role engineering services.

Option C For an outsourced, bottom-up effort, effectiveness also comes into play with the selection of a firm that will provide sufficient staffing and leadership. For a bottom-up process, tools will used for role mining and subsequent analysis. Whether or not collaboration tools are used again will be decided upon by the firm providing the role engineering services.

Option D For an effort that includes both in-house and outsourced staff and both top-down and bottom-up processes, a dual staffing situation will be the case, with the need for making available in-house staff and obtaining other staff from a vendor. Team leadership can come from either the enterprise or the outsourced vendor. One scenario would be that the enterprise furnishes a team leader and the vendor furnishes staffing to conduct the specialized tasks of role mining and preliminary role analysis.

Cost Considerations

Option A For an in-house, top-down effort, cost considerations come into play regarding the compensation of the individuals to be assigned to the team and also the income that may be lost as a result of the individuals' being assigned to the team rather that to a revenue-generating task.

Option B For an outsourced, top-down effort, cost considerations come into play regarding the rates or prices being charged by the vendor. Here the actual labor costs are of no direct concern as the pricing structure will have absorbed the particular costs to be incurred. Since this option is for a top-down approach, there will be some sensitivity in the pricing to the types and number of staff members to be made available to conduct interviews and participate in joint sessions. Unless a fixed price is offered, this variability in staffing could be a concern to the enterprise and affect the scope of the effort.

Option C For an outsourced, bottom-up effort, cost considerations come into play regarding what will typically be a fixed price charged by the vender. Therefore, staffing issues would not be expected to affect this option.

Option D For an effort that includes both in-house and outsourced staff and both top-down and bottom-up processes, the cost consideration come into play as a combination of the other options. For the enterprise staffing, individual compensation and possible loss of income compared to assigning the individual to a revenue-generating task would be the primary factors. For vendor staffing where labor is charged by labor category, there will be a cost sensitivity seen by the enterprise.

Risk Considerations

Each of the staffing options has a set of associated risk factors. Table 13.2 summarizes some of these risk factors.

As Table 13.2 illustrates, each staffing and process option has its own risk factors. For smaller organizations, Option A can serve to get started with role engineering on a relatively limited scale. Once experience has been gained, other options can be considered to extend the role definition results using the same staffing and process or possibly using outsourced staff with or without a bottom- up process. For larger organizations where it is known that existing IT resources contain implicit role definitions, it may be advisable to include a bottom-up process with or without outsourced staff.

Stability Considerations

Option A For an in-house, top-down effort, the sensitivity to team stability is the greatest. The enterprise is solely responsible for the team's care and feeding. Care must be taken to motivate and reward team members to obtain the needed degree of stability to capitalize on knowledge and training and be able to produce quality results readily.

Option B For an outsourced, top-down effort, stability is the concern of the vendor. The enterprise role here is in selecting a vendor whose track record for team stability is acceptable. This can to some extent be determined by soliciting and reviewing past performance data. If a vendor is continually changing personnel, the success of the effort will be greatly jeopardized.

Option C For an outsourced, bottom-up effort, stability is the concern of the vendor. Because the tasks for bottom-up role mining are more technical than are the tasks associated with top-down processes, with their requirements for

Table 13.2
Staffing and Process Options and Their Risk Factors

Option—Type of Effort	Risk Factors	Mitigation Options
Option A: In-house staff, top-down process.	In-house staff lacks know-how regarding role engineering tasks and uses resources inefficiently or ineffectively.	Use an experienced consultant to provide guidance to the effort. Start with a simple effort to provide a learning experience.
Option B: Outsourced staff, top-down process.	Outsourced staff lacks knowledge and understanding of organizational workings and misses or misinterprets key facts.	Use of the top-down process employing the organization's SMEs provides a degree of mitigation. Provide a communication mechanism such as frequent meetings and reviews of interim results.
Option C: Outsourced staff, bottom-up process.	Outsourced staff lacks knowledge and understanding of organizational workings and misses or misinterprets key facts. The bottom-up process is effective in extracting role data from IT resources, but lacks the ability to consolidate role definitions to produce enterprise roles.	Provide input from the organization to the outsourced team using a communication mechanism such as frequent meetings and reviews of interim results.
Option D: Both in-house and outsourced staff and both top-down and bottom-up processes.	In-house and outsourced staff fail to communicate adequately and results are suboptimal.	Establish well-defined lines of authority and communication. Provide a communication mechanism such as frequent meetings and reviews of interim results.

interpersonal skills, it is less likely that stability will be a concern for this option.

Option D For an effort that includes both in-house and outsourced staff and both top-down and bottom-up processes, stability is as of direct concern to the enterprise as it is in Option A and is of indirect concern in selecting a vendor that will provide a stable team, especially in the case of a top-down process.

Team Management Functions

In their book on project management [1], Cleland and Ireland identify a set of functions for the management of a team. They are summarized in Table 13.3.

While these team management functions pertain to teams in general, they apply as well to a role engineering team. It would be easy to overlook these functions and their necessity in managing a role engineering team. Table 13.3 includes statements on the relevance of the team management functions for a role management team. In practice, these team management functions must be translated and refined into specific approaches and acts that will realize the functions on a practical basis.

In addition to the core members of the role engineering team, we recommend including one or more fairly senior managers, even if their available time is limited. There is a strong possibility that the role engineering effort will provide feedback on opportunities or needs to change the way jobs relate to each other or even to change organization structure. In these cases, a manager who is thoroughly familiar with the rationale for such a change could be a real benefit to the effort and possibly enhance the value of the effort to management. Having a manager participate can also promote the best kind of management buy-in.

Team Building

Once staff for a role engineering team has been identified, the individuals must be brought together to act as a team. At a kickoff meeting the goals, objectives, and expectations should be presented and discussed. Each team member should come away from the meeting with a solid understanding of what they are to do and hopefully with a sense of purpose. However, it should not be assumed that the kickoff meeting will suffice for the rest of the effort to induce the team members to work together productively.

The team should meet on a regular basis, preferable once per week. This will facilitate interplay among the team members and will promote communication and cooperation. The team dynamic should be one of a coordinated group effort and not an amalgamation of individual efforts. The power of a team is in its ability to transcend the efforts and results of the members each working in isolation.

Table 13.3
Team Management Functions

Team Management Function	**Description**	**Relevance for a Role Engineering Team**
Team planning	Defining objectives, goals, strategy, and resources for the team.	Outlining the scope of the effort and the process for carrying it out. Resources here refer to software tools for processing and recording role definition results.
Team organization	Defining roles of team members, orienting the team to authority, responsibility, and accountability; ensuring the coordination of the team members and results.	While the role engineering task is primarily one of eliciting policies and requirements and analyzing these, the operation of the team depends on definition of and adherence to lines of authority and responsibility and cooperation in producing results.
Team motivation	Identifying motivating factors, providing an acceptable leadership style, measuring team productivity, engendering a positive atmosphere.	One of the motivating factors that is specific to role engineering is success in gathering information and in processing that information into useful role definition products. The ability to generate these products is a measure of team success and can serve to motivate or demotivate members of the team.
Team direction	Ensuring that the team leader is qualified to lead the team and has an acceptable leadership style and inspires confidence, trust, loyalty, and commitment.	For role engineering the team leader must be well versed in the process and be experienced in leading teams of the type used for role engineering. No one leadership style is required, but a collaborative style is appropriate to small teams such as those conducting role engineering.
Team control	Establishing and applying standards for team members and for the team as a whole, providing feedback to higher management on the performance of the team, reviewing progress with the team, assessing the team's effectiveness and efficiency, ensuring that team members understand how the team operates and how it is controlled.	This function applies to managing any team, but it is important to maintain the productivity and cohesiveness of the team.

If the team can meet face to face, it should do so. The bandwidth for exchanging information available in a face-to-face meeting is much greater than is possible with other types of meetings. Other forms of meetings should be held if face-to-face meetings are not feasible. These can include video teleconferencing and phone calls. Even video teleconferencing has its limitations as to the amount of information than can be exchanged because of its more formal protocol and structured interaction.

The work efforts between phone calls can benefit from the use of collaboration tools such as Web-based tools (for example, WebEx) and peer-to-peer tools (for example, Groove). The primary benefit of using one of these tools will be in maintaining working drafts of work products for the team to share. Depending on the inclination and needs of the particular team, other features of these types of tools, such as mail, scheduling, and joint editing may also be exploited.

Staff Selection

Once the scope of the role engineering effort has been determined, the types and numbers of staff can be estimated. This process should not be rushed because the results will directly affect the success of the effort. For determining the types of staff, the key factors will be the areas to be covered (e.g., mission oriented, administrative, financial), the documentation requirements, the degree of automation of the role engineering processes (e.g., data analysis, storage, display, and publication), and project review requirements. For determining the numbers of staff, the key factors will be the estimated number of permission, roles, and constraints to be identified, engineered, and documented. In general, each area to be covered will require a lead role engineer. In turn, each role engineer can be expected to need two professional support staff which may be either full- or part-time according to the estimated level of effort. The goal regarding staff size is to avoid both understaffing and overstaffing Tables 13.4 and 13.5 summarize these rules of thumb for estimating staff size and duration of the need for staff.

These rules of thumb are provided as a suggested starting point in the planning process. As additional data becomes available, these estimates should be revised to reflect further investigation prior to starting the effort and actual experience once several efforts have been completed.

Table 13.4
Rules of Thumb for Estimating Staff Size

Areas to Be Covered	Types and Numbers of Staff
1	Lead Role Engineer (1)
	Professional Support Staff (2)
2	Lead Role Engineer (2)
	Professional Support Staff (2–4)
3	Lead Role Engineer (3)
	Professional Support Staff (4–6)
4	Lead Role Engineer (4)
	Professional Support Staff (6–8)
5	Lead Role Engineer (5)
	Professional Support Staff (8–10)

Types of Individuals Needed

Now we consider what types of individuals will be needed to staff a role engineering effort and what they will be doing. Typical staff profiles are provided in Table 13.6.

Staff planning should include identification of specific qualifications for the required staff; time phasing of staff availability, review and selection of available candidates, and population of work breakdown structures with staff identification and loading. Consideration should be given to bring new staff into the mix to expose them to role engineering and make them part of the growing pools of available qualified staff.

Corporate commitment to staffing needed for the role engineering effort is, of course, necessary. The impending effort should be described and relevant information disseminated to provide a widespread awareness of the anticipated activities, expected results, and possible involvement of individuals throughout the organization. Financial consideration regarding staffing must be addressed and approved by management prior to beginning any effort. This commitment will rely on accurate estimates of level of effort and associated costs and time frames. While not a direct part of project staffing, commitment of SME resources will also require management commitment, especially since the SMEs are needed to perform their professional duties in addition to their role engineering ones.

Table 13.5
Duration of Staff Need

Amount of Work to Be Done*	Duration of Staff Need (Months)
10 roles 30 permissions 5 constraints	3
25 roles 50 permissions 10 constraints	5
50 roles 75 permissions 15 constraints	6
100 roles 125 permissions 20 constraints	10
500 roles 200 permissions 25 constraints	17
1,000 roles 250 permissions 30 constraints	50

*Number of roles, permissions, and constraints.

Leadership

The qualifications and personal characteristics of a project leader can greatly influence the ultimate success of a project. This implies that the selection of a project manager should not be left to chance or to convenience.

Communications

Exchange of information among the team members for a role engineering project is important, as is exchange of information between the project and enterprise management and stakeholders. Lack of effective communication

Table 13.6
Typical Staff Profiles

Title	**Job Function**	**Duties**	**Comment**
Role Engineer/ Analyst	Collect data Identify permissions Create role names Identify constraints Conduct role mining	Create scenarios Select key elements of scenarios Review existing role definitions (or role "ingredients")	One of these role engineers will serves as team leader
Meeting Facilitator	Prepare meeting agenda Record meeting notes Prepare meeting reports Record action items	Verify scenarios and inclusion/exclusion decisions with SMEs Host and document meetings Assist SMEs in resolving issues Track and assist in addressing action items	The meeting facilitator must be experienced in group facilitation
Data Engineer/ Analyst	Maintain data and analysis results Process raw data to produce permission catalogs, role repositories, constraint catalogs, and hierarchy definitions	Cluster permissions Normalize permissions Model and test constraints Model and test role hierarchies	Data engineers and analysts need experience using generic or specialized tools to process and record metadata and its instantiation
Documentation/ Publication Specialist	Prepare and edit formal reports for project and results Prepare material for posting on Web site(s) Maintain data repositories	Edit and format reports, white papers, correspondence Review and edit role engineering results	This specialist need not have specific role engineering experience but must be adaptable to the role engineering environment
Technical Advisor/ Reviewer	Provide expert advice to staff Provide comments on meeting conduct and results Review project deliverables	Attend selected meetings Comment on project deliverables	Technical advisors and reviewers should have specific or related experience in the role engineering and requirements gathering processes

within and outside the project can seriously jeopardize the success of the current project and perhaps future role engineering projects.

Motivation

Motivation of project staff is an often overlooked factor in staffing and project management. When role engineering is a relatively new activity for the staff members, motivating them to perform well and to persevere through possible difficulties becomes important. Because each organization is to some extent unique, it will be necessary to tailor the basic processes to the situation at hand. This entails the availability of staff members who are self-starters and creative in their work. Further, role engineering cuts across many different knowledge and business domains and involves the associated challenges that these present. Well-motivated staff members are needed to work in these varying environments.

The goal is to motivate role engineering staff to perform to the best of their ability and to contribute to “the learning organization” [2]. In [3], Cleland and Kocaoglu present a set of job motivational factors based on responses by professionals in engineering organizations to a questionnaire containing 29 items that relate to motivating the individual to do his or her best work. Each respondent rated the top five factors and the most frequent responses were identified. These results are provided, in alphabetical order, in Table 13.7.

All of these factors should be considered important for members of a role engineering team. We recommend also that the team leader discuss the factors with the team to prioritize the factors, perhaps add new ones, and also to communicate with the team members on how these will be realized.

Staff Development

For a first in-house role engineering effort, staffing will necessarily come from a selection of experienced onboard individuals or new hires. Thus, the staff selection process is important at this initial stage. Once a role engineering effort is underway, there will be opportunities to develop additional staff to enlarge the pool of available role engineers. Here individuals with appropriate academic preparation and work experience can be assigned to a role engineering team to assist and to learn its techniques and communication methods. In addition to this on-the-job training, specialized training in role

Table 13.7
Motivational Factors for Professionals in Engineering Organizations

Chance for promotion
Chance to turn out quality work
Feeling my job is important
Getting along well with others on the job
Good pay
Large amount of freedom on the job
Opportunity for self-development and improvement
Opportunity to do interesting work
Personal satisfaction
Recognition by peers
Respect for me as a person

engineering techniques can be conducted in house or be outsourced. Over time the additional staff can be assimilated into the team and, as desired, new staff can be similarly trained.

Concerning whether or not staff members may need any additional training, generally speaking the customary IT staff training and experience should suffice to a great degree. For the data engineer/analyst, data modeling training and experience such as that needed for data and database administration will suffice as a solid starting point. The meeting facilitator will need to be briefed by the role engineer(s) on the goals and objectives of the role engineering process. The role engineers themselves should read this book and also obtain on-the-job training with similar projects if possible.

Staff Evaluation

As part of the process of ongoing project evaluation for purposes of continuous improvement and justification to management, staff performance and related matters should be recorded and communicated both within the team and to enterprise management. In addition to the customary performance criteria for employee evaluation, the role engineering team criteria can include the following records:

- Attendance at meetings and teleconferences;
- On-time completion of analytical assignments;

- Thoroughness in preparing meeting reports;
- Initiative in preparing and presenting innovative approaches;
- Willingness to mentor less experienced staff;
- Leadership on assigned tasks.

These criteria and any others in use should be discussed with existing and new staff to permit them to perform to the criteria and to self-evaluate prior to formal evaluation by others. Those meeting or exceeding the criteria should be appropriately rewarded financially and through promotions and benefits. Those failing to meet the criteria must be considered for elimination from the team. This will be necessary at some point in time, whether or not replacement personnel are available. For this reason it is advisable to have a process for making qualified personnel available to the team, using in-house development or acquisition of new staff.

Staff Retention

Closely related to motivation, development, and evaluation of project staff is retention of the team members over time. One of the reasons for the importance of staff retention is that the knowledge and facility that the staff members acquire while carrying out the role engineering process can make the team very efficient in defining good roles. This valuable resource should be maintained to the greatest extent possible. An effective reward structure can promote this desirable force for retention of team members. Recognizing the motivational factors listed in Table 13.7 within the project should promote retention of this valuable resource.

References

[1] Cleland, D. I., and L. R. Ireland, *Project Management*, New York: McGraw-Hill, 2002.

[2] Senge, P. M., *The Fifth Discipline: The Art and Practice of the Learning Organization*, New York: Currency Doubleday, 2006.

[3] Cleland, D. I., and D. F. Kocaoglu, *Engineering Management*, New York: McGraw-Hill, 1981.

14

What Can Go Wrong and Why?

Despite its title, this chapter is a positive one because it is intended to contribute to the success of a role engineering effort by helping to avoid some common pitfalls. These have been largely identified from first-hand experience with several role engineering projects. The potential pitfalls fall into two categories:

1. Quality of role definitions;
2. Problems in execution of the role engineering process.

Avoiding these pitfalls will require awareness in planning and carrying out the role engineering effort.

Quality of Role Definitions

Access Control Policy

First, good role definitions depend on the availability of an accurate access control policy. This policy should be defined prior to creating role definitions. For a bottom-up approach, existing permissions and implicit roles will embody the access control policy. However, in practice this de facto access control policy will typically need to be refined and verified. A clean-up phase must be undertaken in most organizations to weed out over-assignment of permissions to individuals and, by way of the engineering process, to roles. For a top-down process the definition of the access control policy will be part of the role engineering process. This is because the access control policy will include the role names that are defined during the role engineering effort. A third possibility for this needed access control policy would pertain when using a top-down approach. Here the participants in the role engineering process, both subject matter experts and IT staff, mentally recognize elements of the access control policy among the permissions and the role names as these are defined. That is, the participants use their internal mental models of the policy and accordingly make role design decisions. The advantage of this approach is that it avoids a potentially difficult and time-consuming process of defining the access control policy. Disadvantages include subjectivity and the fact that the access control policy is not documented. This in turn introduces corresponding subjectivity into the process of verifying role definitions against the access control policy.

Inadequately Engineered Roles

Good role definitions can greatly benefit the enterprise by simplifying the management of permissions, promoting the principle of least privilege, and facilitating the use of hierarchies of permissions and the implementation of constraints of various types. Just as good role definitions can benefit the enterprise in various ways, less-than-good role definitions can have the opposite effect. Thus, for example, the consequences of defining inadequate roles can lead to misassignment (either over- or under-assignment) of permissions to users in violation of access control policy or the principle of least privilege. Other ways that inadequate roles could violate access control policy could be in relation to hierarchies and constraints and their possible interactions. If hierarchies and constraints are improperly designed, the resulting roles could fail to reflect the enterprise realities (job functions, lines of authority, capabilities) and access control policy.

Chapter 5 addresses the topic of defining good roles. This implies that it is also possible to define bad or at least inferior roles. This chapter addresses that possibility and how to avoid it. Another possibility is that execution of a role engineering process can run into some pitfalls that could waste resources, lead the role engineering team into a dead end regarding the maintenance of role definitions, or make progress at too slow a rate and therefore not achieve the desired results within a useful time frame.

What can happen if roles are not engineered properly? As with any IT activity, there is always a potential for falling into traps and making some common mistakes. These can result in suboptimal or incomplete results, waste of resources, discouragement, adverse effects on RBAC adoption and use, and a poor reputation for RBAC. It is always preferable to avoid a problem than to try to solve a problem, so let us review some potential pitfalls in the role engineering process and point out some ways of avoiding them.

Role Names

First let us begin with the assignment of names to roles. Role names could, for example, be difficult to recognize or even misleading. As mentioned earlier, role names should be easily recognizable and understood by the administrative personnel who will be assigning users to roles. If this cannot be done with reasonable ease, the administrative advantages of RBAC will be diminished. If an administrative person is required to research the meaning of role names, and then to determine whether a given individual should be assigned to a particular role, a waste of time and effort will result.

In addition to being easily recognizable, role names should be distinct and not easily confused. For example, use "Payroll Clerk" and "Purchasing Clerk" instead of something like "Clerk_1" and "Clerk_2." To keep role names standardized throughout the enterprise, it is not advisable to include references to organizational units or geographical locations in role names. When defining roles using a bottom-up approach, these types of role names are likely to be produced. If they are, an effort to generalize the role names to the enterprise should be conducted.

Permissions

Next, we address the definition of permissions and also how the permissions are assigned to roles. A role can provide excessive or inadequate access to resources, or permissions assigned to roles could be inconsistent. This was discussed in Chapter 5. Here it will be necessary to have a means of testing each role for its intended access permission or denial. This relates to the access control policy that is to be enforced. The policy for each role should be described by those actions on objects that are allowed and those that are not allowed. This policy can be represented by a matrix of roles versus permissions. Figure 4.2 illustrated a role that implements a security policy. By modeling roles as discussed in Chapter 10, the policy can be compared to the modeled roles.

Permission review and auditing should be an ongoing process, since policies and their components change over time. Personnel assignments to roles also change fairly rapidly and must typically be reported upon to support security audits.

Constraints

Constraints on roles could be excessive or inadequate. Just as permissions must be tested for their access or denial properties, constraints must be similarly tested for their effects. Here, again, it will be necessary to have a means of testing each role's constraints for their intended contribution to access permission or denial.

Hierarchies

Role hierarchies could provide excessive or inadequate access to resources where the individual roles did not. The possible complexity of hierarchical inheritance relations among roles is exceedingly great. For example, the combined permissions of two junior roles could provide a set of permissions to a senior role that is more than intended in the access control policy. We note that this situation can occur whether or not the role structures are restricted to tree structures. If unrestricted structures are implemented, this situation can be exacerbated, as then the junior roles can be totally unrelated. It is thus necessary to trace the inherited permissions of each role in a hierarchy that will be assigned to a user. The potential for excessive complexity argues for keeping the use of role hierarchies to a minimum.

Number of Roles

It is possible to define an excessive number of roles for an organization. The existence of too many roles could diminish the ability to understand and manage the roles and to assign users to the roles. Where a large number of roles are warranted, it will be important to provide subsets of roles to those entrusted with managing role definitions and with assigning users to roles. This can mitigate the problems caused by the large number of justifiable roles for the organization.

Problems in Execution of the Role Engineering Process

Beyond the potential pitfalls related to role definition, additional pitfalls related to execution of the process should also be avoided. To help you avoid some of these, we describe some DOs and DON'Ts and also provides some suggestions for foreseeing and avoiding problems.

One of the most prominent potential pitfalls in a role engineering effort is trying to accomplish too much at a time. If the scope of a role engineering effort is overly ambitious, the result can be a failure to accomplish even the simplest goals of the effort. The KISS (keep it simple stupid) principle should be adhered to in these projects as in most projects. This principle applies both to top-down and bottom-up approaches.

For example, when a role engineering effort is being justified and then planned, it may be tempting to try to include the full range of RBAC features. In addition to defining a set of structural roles, which by themselves could provide significant improvements to the enterprise, it may appear desirable to include some functional roles as well. Or, if an effort is to address functional roles, it may be sufficiently ambitious to define a set of core RBAC roles and defer the implementation of constraints or hierarchies to a later effort. Even defining a set of core roles can present a sufficient challenge, with its need to define the access control policy, determine role names, and to identify permissions.

Efficiency in the Use of Role Engineering Resources

One area noted in the VHA experience was that of keeping SMEs focused on what they do best—that is, to identify, classify, split, combine, and remove various candidate artifacts. Extraneous matters, such as what tools to use, how the existing system works (as a criterion), how they think about a domain outside their expertise functions, should be noted in meeting notes but passed over to address the most relevant issues that SMEs can successfully address. Additionally, the VHA experience with assignment of work to team members showed that SMEs seldom performed work outside of the scheduled meetings. Therefore, "homework" assignments for a subsequent meeting should probably be avoided, as the expectation of having the homework completed is not realistic and can cause delays. Whenever feasible, the dedicated project staff should perform whatever work is required and then the results can be presented to the SMEs for their review and comment.

Innate Conflicts

Role engineering is not a science and its results are not provably correct. It can be expected that during the process of defining roles that trade-offs and compromises will need to be made. One source of potential innate conflict can be seen from the RBAC models themselves that are presented in the RBAC standard. When we advance from core RBAC to either constrained

RBAC or hierarchical RBAC, there should be little problem. However, if it is decided to combine constrained RBAC with hierarchical RBAC, certain potential conflicts are predictable. Research has shown [1] that when certain designs combine hierarchies and constraints, the results can contradict the access control policy. For example, if a senior role inherits two junior roles and one of these is constrained by a separation of duty constraint and the other is not, the inherited result could allow circumvention of the separation of duty constraint by the senior role. In such cases it may be necessary to simplify the hierarchy structure, the constraints, or both. It is advisable to model the operation of the system of roles to ensure that only the desired consequences are produced by the design. This step can entail an added expenditure of time and resources.

Maintenance Planning

In principle, roles can be engineered to implement all and only the desired security policy. As with designing software and other artifacts, there will often be more than one way to design a given role artifact. Also, the design can be expected to evolve, such that new potential objects and relationships will come into play. As a result, as analysts attempt to incorporate current policy into an RBAC system, it will be necessary to solve additional problems relating to role design. Therefore, this ongoing maintenance of role definitions should be planned and budgeted.

Backtracking

When it becomes necessary to update the design of a role structure, we may find that earlier decisions have hindered or precluded the optimal design that might otherwise be possible. With this in mind, it may be possible to foresee some of the future states that may be hindered by past decisions and to consciously avoid the potential problems. Here we identify some possible role definition decisions that could adversely affect the range of possible role designs at a future date.

A simple example of a role definition decision that can render subsequent role definition difficult is when a role is assigned a particular permission that should not have been assigned to that role. Perhaps this permission was quite dissimilar to the other permissions assigned to the role or the operation on the object is a special case that is hard-coded into an application. If at a future time it is desired to remove this permission from the role definition, two different problems can be created. First, once the permission is removed, the users assigned to the role will be unable to perform certain functions in the system. This condition would be propagated to any senior roles that inherit the changed role's permissions. The other problem would be that the removed permission may be assigned to a different role or perhaps to a new role. In this case, the receiving role could provide an excessive amount of access to the users assigned to the receiving role.

Another example of a decision that could require backtracking would be in the selection of a role name that is too specific. If the name reflects such transient characteristics as organizational unit or geographical location, it may be necessary at a later date to change these to reflect changes in organization or location. In other cases of overly specific roles names it may be necessary to change these to be more general and thus reflect the access control policy in a more general way. When role names are changed it is advisable to establish a mechanism, possibly using inheritance, whereby more than one role name carries the same set of permissions. This would avoid the need to delete one role definition and replace it with another and thus disrupt the functioning of the RBAC environment.

A further example of a class of decisions that could require backtracking would be where a bottom-up approach is taken and role definitions are deduced from permissions present in existing systems. As is well recognized, these role definitions can turn out to be too reflective of the systems from which the definitions were derived. One way to remedy this situation would to be to define more senior level roles and to have these inherit the permissions of the overly specific roles.

Other Limitations of Role Engineering

When performing bottom-up role identification, different organizations can have different definitions of what a role is. This does not to refer to definitions of particular roles, which is not too surprising, but rather to definitions of what a role is. The conflicts that ensue from these differences must be resolved in some manner if an enterprise role structure is to be defined effectively. Effectively, coming to an understanding on the definition of the term "role" would be part of adopting and tailoring a role engineering process to the situation at hand. Once the role engineering process has been established for the effort, all parties involved in the role engineering process must be made aware of this definition and the remaining ones included in the role engineering process.

Overcoming Obstacles

An ounce of prevention is worth a pound of cure. This book is intended to assist in the prevention of some common or occasionally less-known pitfalls in the conduct of role engineering efforts. The time to avoid these pitfalls begins with creating a plan for the effort that allows for the required analysis and synthesis that takes into account both positive and negative measures that lead directly to the achievement of the role engineering goals. During the execution of the plan, for the role definition process itself and in review meetings, the guidelines for defining good roles and avoiding inadequately defined roles should be kept in the foreground and included in work definitions and in review criteria.

Practical Guidance from Eurekify, Ltd.

This information provides additional guidance and tips from role engineering practitioners at Eurekify.

Since different types of roles are going to be used by different people, it is important to take that into consideration.

- Business roles will mostly be presented to line-of-business managers (e.g., to assign to new recruits, to recertify as part of compliance processes, and so forth).
- Applicative and IT roles will likely be used by resource owners and administrators, and their names should capture the distinct functions that they allow, and if the roles are defined within a multiapplication platform (e.g., Microsoft Active Directory [AD]), then they should also capture the name of the application.
- Applicative and sometimes business roles will also be presented to auditors when they look for violations of separation of duty (SoD) and other business process rules, so their names and descriptions should also capture the essence of the business function they allow.
- Some resource roles are related to a specific deployment, server machine, or geographical location or responsibility and should thus capture those.

- In some cases, role names should preferably reflect the provisioning rule based on which they are granted, because this would allow quickly identification of them when needed or identification of incorrect assignments within them (e.g., when a person leaves a position and is not removed from the role).

The bottom line is that in naming a role, you should always balance between names that support the most typical current use of the role and names that will be stable for the long term. Always think of the users of the role and how the role name will help them make quicker and correct decisions without having to dive into the role more specific attributes.

One of the challenges in role engineering is to use the right level of granularity that would provide sufficient control, while at the same time not result in an administration nightmare. Here are two simple tips:

- If two or more (sometimes many) roles are only distinguished by a few permissions, and if these permissions are not too sensitive, consider merging them into a single role. You will be granting some users a few more privileges than they really need, but the administrative gains will be substantial. Automated tools can easily identify these situations and guide you through the decision.
- If two or more (sometimes many) roles share a certain set of permissions ("base"), consider creating another role to represent this base, and take those permissions out of the other roles. If you prefer, you may also use a hierarchical representation in this case. While adding one more role, this will again simplify administration substantially. Again, automated tools can do this for you.

There are more ways in which you can substantially simplify a roles structure.

You may want to distinguish two types of constraints or rules.

1. Constraints that represent operational considerations (such as provisioning);
2. Constraints that represent audit requirements (such as SoD).

Provisioning constraints should be specified as part of the role's rule and shall be created in a way that makes it easy for the provisioning engine to use them. Different provisioning engines use different languages, some more and less expensive. Furthermore, performance may be affected depending on the underlying infrastructure (e.g., database versus directory).

Audit constraints should usually be separated so that they are more easily managed and adapted by auditors. The language used shall again be suited to auditors (e.g., separation of duty rules and other business process constraints).

Another constraints and rules consideration is the time when they shall be applied. Auditors are traditionally testing against such rules in an offline fashion, as part of the periodic IT controls review (twice a year on average). However, a new set of tools allows for real-time testing just before a privilege is being granted in a so-called "preventive" fashion. There are limitations to both approaches to testing, and the types of checks that can be done in real time without hampering the provisioning process is also limited.

Role hierarchies present several challenges, including being counter-intuitive and being subject to high complexity. The semantics of a role hierarchy can often be implemented in a flat structure, for example, by grouping together common access rights into an additional specially crafted role and then assigning this role separately to the users of the intended parent roles. Some automated tools will help you "flatten" a hierarchical role structure and/or to create a hierarchy.

To know whether the number of roles is excessive, consider the line-of-business manager that is required to operationally approve role definitions and role assignments. While a single role's administrator can be responsible for more than 1,000 roles, the line-of-business (LOB) manager should not be concerned with more than 20–50 roles on average. In some organizations we find an LOB manager accountable for 500 roles, but then delegating the administrative responsibility to other LOB managers for the daily operations.

In practice, we find more organizations defining fewer roles with less coverage than needed. As a rule of thumb, role definitions shall cover 80% of the overall privileges, covering more than just the most common privileges. Many deployments of identity management often fall short of this mark and define only 50–100 very simple roles. Of course, such practice does not result in any significant administrative gains.

It is important to note that roles should only be defined if they can be relatively stable over time. Also, roles should be defined to capture the rule, and not the exceptions. The number of roles should be controlled, as explained earlier. Finally, some privileges are temporary, specific to a person but not his or her role and so forth. For this reason, we prefer to leave some privileges (~20%) as ad hoc rather than role-based.

Reference

[1] Schaad, A., J. Moffett, and J. Jacob, "The Role-Based Access Control System of a European Bank: A Case Study and Discussion," *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, Chantilly, VA, 2001, pp. 3–9.

15

Summary and Conclusion

We have covered the key areas of the role engineering process for the purpose of assisting you in approaching and conducting a successful role engineering effort. The goal of role engineering to benefit the enterprise can be met by following the guidance provided here. This guidance should help you get started efficiently and keep your effort on the right track. Your effort will be effective in producing good role definitions and productive in its use of resources. In this final chapter we review the key areas we have examined and we provide a few additional recommendations.

In reviewing what we have covered, we summarize the key areas and ideas relevant to role engineering. These include getting started, producing role definitions, solving problems, and taking advantage of what others are doing.

Making the Business Case

Unless there is a business case to undertake a role engineering effort, there is no need to go any further along that path. Practically speaking, making the business case will be necessary for obtaining approval and a commitment of resources to a role engineering effort. The business case for role engineering is actually part of the business case for RBAC itself. When justifying RBAC for adoption by an organization, the costs and benefits for role engineering must be factored into the RBAC business case. Some of the justification for RBAC and role engineering may go beyond the requirements of the mission of the enterprise and how these requirements are carried out. One example of these requirements is external mandates for confidentiality, integrity, and availability. Of course, the enterprise itself may benefit from these as well. Another example is the potential of RBAC in implementing constraints such as separation of duty, which may benefit an external organization for which the enterprise is performing a service or supplying products.

Part of the business case will include cost-benefit analysis. Obtaining data on both costs and benefits is a challenge, although benefits tend to be more of a challenge to determine than costs. This is because benefits tend to include intangible items such as avoiding difficulties and conforming to mandates. As time goes on it will be possible to refine estimates based on experience. Therefore, it is advisable for every role engineering effort to collect data on costs and benefits for replanning current efforts and for justifying and planning future ones.

Integrating Role Engineering into the System Development Life Cycle

As stated in Chapter 3, a role engineering effort may be undertaken as an independent effort at the enterprise level or as part of a system development project. When it is undertaken within a project, there must be a means of generalizing the results to the enterprise as a whole. This presupposes an enterprise model and a repository of role definitions. Then each project can add its individual updates to the repository in a consistent manner. Even when role engineering is performed at the enterprise level, there will typically be a need for the addition of role definitions that were not covered. This means that all parts of the organization must be aware of the enterprise role engineering activities and results and be provided a mechanism to forward their contributions for review and consideration for inclusion in the repository.

Defining Good Roles

Good roles are ones that have names that are readily recognized by administrative personnel and have permissions, constraints, and hierarchies that accurately reflect the access control policy. Roles may be developed in various subject domains, such as business, finance, administration, and security, and may be either structural or functional. Structural roles have permissions that permit connection or access to a gross level IT resource, such as a network, server, workflow, or device; and functional roles have permissions that control finer-grained accesses within an IT resource.

There must be a balance between roles with excessive numbers of permissions, which can violate the security principle of least privilege, and roles with too few privileges, which can obviate the advantage of RBAC that the number of roles is much less that the number of privileges. When the number of roles is excessive, role administration begins to approach the same level of complexity as assigning individual privileges to users. Good roles will accurately reflect the access control policy. This implies that the access control policy, in terms of users and permissions, is known and documented. Defining permissions necessitates knowing what the protected objects and allowed operations are, as well as recognizing any constraints among users, roles, and permissions.

The Process of Defining Roles

The role engineering process revolves around the enterprise job functions, responsibilities, organizational positions, and authorities that are considered to be relevant for access control. To avoid wasted effort and creating inadequate role structures, a defined process should be used to identify role names and associated permissions. This process may use a top-down, bottom-up, or combined approach. There are pros and cons to either approach, chiefly in that top-down tends to be more labor-intensive but produces a more general role structure and bottom-up is more automated but can produce roles that reflect the individual applications from which they are derived rather that the enterprise as a whole.

Tools That Can Assist in the Role Engineering Process

Tools can assist the role engineering process but cannot fully automate it. Human skill and judgment are still needed. Acquisition of role engineering tools should be done on the same cost-effectiveness basis as for other acquisitions. Tools expressly designed to perform role engineering functions tend to follow the bottom-up approach, although some of them do perform top-down functions. In addition to tools that assist in the role engineering process are repositories that can assist in the storage of role definitions and their maintenance.

Activities of Organizations Relevant to Role Engineering

A number of role engineering efforts are underway or have been completed. We can learn both positive and negative lessons from these. Perhaps the most fundamental lesson for a successful effort would be to strive for simplicity. It is always preferable to have a small success than to have a large failure. And, of course, an effort with limited scope can be expanded later, once success has been secured.

Other activities relevant to role engineering are academic research and standardization efforts. Over time, academic research can provide support for role engineering processes and can inject new ideas and approaches into how roles are constructed. First, the results of academic research must be understood and applied to solving problems. Because of the richness in the combination of business needs and access control requirements, it is likely that academic research relevant to role engineering will continue for the foreseeable future.

Standardization efforts relevant to role engineering can be found in defining the components of an RBAC system and in role content itself. Perhaps the primary examples of these categories are the ANSI INCITS RBAC Standard (RS) [1], the OASIS XACML RBAC Profile [2], and the HL7 standard permissions for healthcare [3]. The RBAC Standard provides a reference model for roles that can be used to define instances of roles for an enterprise. This book has made use of this model in describing the process of role definition. The OASIS XACML RBAC Profile deals with RBAC in the increasingly important area of Web services. The HL7 standard permissions, while they constitute a special case that applies only to healthcare, serve as an example of how such definitions can be created and promulgated. The HL7 permissions also reflect the fact that standardization of permissions is feasible, albeit at a rather high level of abstraction, even though it is not feasible to standardize roles across enterprises.

Planning and Staffing a Role Engineering Effort

Planning a role engineering effort includes justifying and securing management approval for it. Therefore, the proposed effort must be described and its return on investment must be estimated. Cost-benefit analysis should be conducted, and this implies that costs and benefits can be quantified or at least characterized. Risks must be identified and their mitigation anticipated. Additional requirements include establishing goals, adopting strategies and methods, accomplishing staffing, and setting up control mechanisms for use as the effort is carried out.

Scoping the effort is important to its success. The following set of objectives can be considered for inclusion in a proposed role engineering effort:

- Define candidate roles;
- Identify constraints;
- Prepare a permissions catalog;
- Define enterprise roles;
- Assign permissions to enterprise roles;
- Apply constraints to role definitions;
- Design hierarchies among roles.

Staffing estimates must be verified, and both the staffing and funding estimates must be expanded into executable plans for conducting the effort. These plans should include a communication plan that will foster mutual awareness and cooperation among the participants.

The plans should also include collecting data for use in future activities, including the following:

- Justification (e.g., cost-benefit);
- Estimation (updates to resource rules of thumb);
- Continuous improvement;
- Reference by others.

The collection of this performance data during the conduct of the effort is aimed at closing the loop on the planning and execution cycle for role engineering.

Potential Pitfalls and How to Avoid Them

The potential pitfalls fall into two categories: the quality of role definitions and problems in execution of the role engineering process. Avoiding the pitfalls can be accomplished by the following activities:

- Scoping the effort to realistic objectives;
- Defining an accurate access control policy;
- Naming roles using readily recognized names;
- Balancing the relative number of permissions assigned to each the roles;
- Testing role definitions against the access control policy;
- Judicious use of constraints and hierarchies;
- Making most beneficial use of available staff, including subject matter experts.

Reminders of Key Recommendations

The following items could be used as a check list for use when planning a role engineering effort:

- Make the business case by estimating return on investment;
- Identify risks;
- Use a defined process to identify role names and associated permissions;
- Collect data on costs and benefits;
- Generalize the results to the enterprise when conducting role engineering within a project;
- Use role names that are readily recognized by administrative personnel;
- Balance the number of permissions across roles;
- Acquire role engineering tools only if they can be justified on cost-effectiveness;
- Strive for simplicity;
- Follow standards where possible and appropriate;
- Capitalize on the experience of others.

What We Can Expect in the Future

One expected development is in the area of standards. These include standards regarding RBAC infrastructure and those regarding role engineering. The existing RS has proven to be useful for discussing RBAC implementation in the infrastructure sense. Its models and definitions provide guidance to system designers on commonly accepted concepts and assumptions. Complementing the RBAC Standard is the draft RBAC Implementation Standard [4] that is intended to narrow down the possible interpretations of the RS and add specifications for interoperability and audit. While standardized roles are not feasible since each organization's roles are different, it is possible to standardize permissions at the business process level. This has been done for healthcare, initially by the Veterans Health Administration, and now in the Health Level 7 organization. HL7 is also developing a standard role engineering process.

The benefits of standardization are several. Standards make it simpler to compare two or more implementations side by side and evaluate them. They can also promote interoperability among systems. Standards for role engineering can foster a community of distributed efforts to define roles or components of roles. By conforming to a common standard, the results of these distributed efforts can be combined and reused.

Since RBAC is still relatively new in actual implementations, it has been necessary for people to change their mental models of security and its administration. Many compromises are required as RBAC is retrofitted into existing system architectures. This is, of course, a predictable state of affairs. Of course, some RBAC features are being designed into systems. At present there is still only limited support for a full RBAC solution. For example, much of the RBAC seen in commercial products supports structural roles but there is little support for functional roles. Some of this support for functional roles will depend on the availability of applications that are RBAC aware and that can use an external policy decision point to control access within them.

Along with development of more sophisticated RBAC products must come a corresponding evolution in the way people think of access control and administration of security authorization. This is what we might call *RBAC thinking*. There must be a paradigm shift from today's understanding of how current IT systems do things, and then translating that into how RBAC would do it, to seeing access control in terms of RBAC initially, and then tailoring the current IT systems to support RBAC.

A future level of RBAC maturity would see systems, including applications, designed with RBAC in mind such that it is no longer necessary to use the imagination to reveal how RBAC can be realized in a more conventional system architecture.

Final Recommendations

Advancements in role engineering depend on a community of mutual interest and on making isolated results available to the community. Thus, participation in industry sector forums can bear fruit for this interchange of experience and results. Standardization is essential for this to be fully successful. While standardization efforts are known for their relatively long time scales, participation in standardization efforts will help move the various standards through their life cycles, and provide input that will make the standards more useful and usable. Of course, following available standards is required to obtain their full benefit.

Ideally, role engineering will be integrated into enterprise endeavors as a whole and not be considered out of the ordinary. It will fit into requirements engineering and management, development of security policy including access control policy, and business process engineering. In some cases it will be integrated into system development life-cycle models. When role engineering has been accomplished for the enterprise, its principles and processes will form an ongoing process of role maintenance.

References

[1] InterNational Committee for Information Technology Standards (INCITS), ANSI INCITS 359-2004 American National Standard for Information Technology—Role Based Access Control, February 3, 2004.

[2] OASIS, XACML Profile for Role Based Access Control (RBAC), Committee Draft 01, February 13, 2004, http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf.

[3] HL7 Healthcare Scenario Roadmap v.2.20, September 6, 2006, http://www.va.gov/RBAC/documents.asp.

[4] InterNational Committee for Information Technology Standards (INCITS), *Proposed American National Standard for Information Technology—Role Based Access Control Implementation Standard,* Draft, 2007.

Bibliography

Research Papers

Chadwick, D. W., and A. Otenko, "The PERMIS x.509 Role Based Privilege Management Infrastructure," *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, Monterey, CA, June 3–4, 2002, pp. 135–140.

Chandramouli, R., "A Framework for Defining an Access Control Service for Healthcare Information System Using Roles," *Proceedings of the 4th ACM Workshop on Role-Based Access Control*, Fairfax, VA, October 28–29, 1999.

Coyne, E., "Role Engineering," *Proceedings of the 1st ACM Workshop on Role-Based Access Control*, Gaithersburg, MD, November 30–December 1, 1995, pp. I-15–I-16.

Crook, R., D. Ince, and B. Nuseibeh, "On Modelling Access Policies: Relating Roles to their Organisational Context," *Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005.

Epstein, P., and R. Sandhu, "Engineering of Role/Permission Assignments," *Proceedings of the 17th Annual Computer Security Applications Conference*, New Orleans, LA, December 10–14, 2001.

Eurekify, Inc., *Role Management Is Fast Becoming a Required Aspect of Identity Management*, Raanana, Israel, February 2006.

Ferraiolo, D. F., and J. F. Barkley, "Comparing Administrative Cost for Hierarchical and Non-Hierarchical Role Representations," *Proceedings of the 2nd ACM Workshop on Role-Based Access Control*, Fairfax, VA, November 6–7, 1997.

Ferraiolo, D. F., and D. R. Kuhn, "Role Based Access Control," *15th National Computer Security Conference*, Baltimore, MD, 1992, pp. 554–563.

Gallaher, M. P., A. C. O'Connor, and B. Kropp, *The Economic Impact of Role-Based Access Control,* Planning Report 02-1, National Institute of Standards and Technology, 2002.

Hu, V. C., et al., "Conformance Checking of Access Control Policies Specified in XACML," *Proceedings of the 1st IEEE International Workshop on Security in Software Engineering (IWSSE 2007)*, Beijing, China, July 2007.

Kang, M. H., et al., "Access Control Mechanisms for Inter-Organizational Workflow," *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, Chantilly, VA, 2001, pp. 66–74.

Kern, A., and C. Walhorn, "Rule Support for Role-Based Access Control," *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT 2005)*, Stockholm, Sweden, June 1–3, 2005, pp. 130–138.

Kern, A., et al., "Observations on the Role Life-Cycle in the Context of Enterprise Security Management," *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, Monterey, CA, June 3–4, 2002, pp. 43–51.

Martin, E., T. Xie, and T. Yu, "Defining and Measuring Policy Coverage in Testing Access Control Policies," *Proceedings of the 8th International Conference on Information and Communications Security (ICICS 2006)*, Raleigh, NC, December 2006, pp. 139–158.

Martin, E., and T. Xie, "A Fault Model and Mutation Testing of Access Control Policies," *Proceedings of the 16th International Conference on World Wide Web, (WWW 2007),* Banff, Alberta, Canada, May 2007, pp. 667–676.

Neumann, G., and M. Strembeck, "An Integrated Approach to Engineer and Enforce Context Constraints in RBAC Environments," *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003)*, Como, Italy, June 2–3, 2003, pp. 65–79.

Neumann, G., and M. Strembeck, "A Scenario-Driven Role Engineering Process for Functional RBAC Roles," *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002),* Monterey, CA, June 3–4, 2002, pp. 33–42.

Sandhu, R., et al., "Role-Based Access Control Models," *IEEE Computer*, Vol. 29, No. 2, February 1996, pp. 38–47.

Sandhu, R., D. F. Ferraiolo, and D. R. Kuhn, "The NIST Model for Role Based Access Control: Towards a Unified Standard," *Proceedings of the 5th ACM Workshop on Role Based Access Control*, Berlin, Germany, July 26–27, 2000.

Schaad, A., J. Moffett, and J. Jacob, "The Role-Based Access Control System of a European Bank: A Case Study and Discussion," *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, Chantilly, VA, 2001, pp. 3–9.

Schimpf, G., "Role-Engineering—Critical Success Factors for Enterprise Security Administration," *16th Annual Computer Security Applications Conference Issues 2000:*

Enterprise Security Management Special Workshop, New Orleans, LA, December 11–15, 2000.

Science Applications International Corporation (SAIC), *Role-Based Access Control (RBAC) Role Engineering Process*, Version 3.02, San Diego, CA, January 20, 2005.

Books

Cleland, D. I., and D. F. Kocaoglu, *Engineering Management*, New York: McGraw-Hill, 1981.

Cleland, D. I., *Project Management*, Blue Ridge Summit, PA: Tab Professional and Reference Books, 1990.

Cleland, D. I., and L. R. Ireland, *Project Management*, New York: McGraw-Hill, 2002.

Ferraiolo, D. F., D. R. Kuhn, and R. Chandramouli, *Role-Based Access Control*, 2nd ed., Norwood, MA: Artech House, 2007.

Harvard Business Review, *On the Business Value of IT*, Cambridge, MA: Harvard Business School Press, 1999.

Jones, C., *Estimating Software Costs: Bringing Realism to Estimating*, 2nd ed., New York: McGraw-Hill, 2007.

Senge, P. M., *The Fifth Discipline: The Art and Practice of the Learning Organization*, New York: Currency Doubleday, 2006.

Standards

ASTM E 1986–98 Standard Guide for Information Access Privileges to Health Information, approved October 10, 1998, November 1998.

Ferraiolo, D. F., et al., “Proposed NIST Standard for Role-Based Access Control,” *ACM Transactions on Information and System Security*, Vol. 4, No. 3, August 2001, pp. 224–274.

HL7 Healthcare Scenario Roadmap v. 2.20, September 6, 2006.

InterNational Committee for Information Technology Standards (INCITS), ANSI INCITS 359-2004 American National Standard for Information Technology, *Role Based Access Control*, February 3, 2004.

InterNational Committee for Information Technology Standards (INCITS), Proposed American National Standard for Information Technology, *Role Based Access Control Implementation Standard*, Draft, 2007.

International Standards Organization, *Health Informatics—Functional and Structural Roles*, Draft Standard for Comments, TC 215/WG4/N214, ISO/PDTS 21298, January 15, 2004.

National Institute of Standards and Technology "Security Considerations in the Information System Development Life Cycle," NIST Special Publication 800-64 REV. 1, June 2004.

OASIS, XACML Profile for Role Based Access Control (RBAC), Committee Draft 01, February 13, 2004, http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf.

Web Sites

Business Process Reengineering, http://www.bitpipe.com/rlist/term/Business-Process-Reengineering.html.

Eurekify, http://eurekify.com/customers.case.studies.asp.

IEEE EWH, http://www.ewh.ieee.org/cmte/pa/Status/Engineering.html.

NIST-RBAC, http://csrc.nist.gov/rbac/.

NIST-RBAC/EDAC, http://csrc.nist.gov/rbac/edac.html.

OASIS XACML, http://www.oasis-open.org/committees/xacml/faq.php

Proposed NIST Standard for RBAC, http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf.

VHA-RBAC, http://www.va.gov/RBAC/documents.asp.

About the Authors

Edward J. Coyne is a senior security engineer at Science Applications International Corporation. His areas of expertise encompass computer security architecture, program management, security certification and accreditation, secure software engineering processes and methods, requirements engineering, IT strategic planning, performance modeling, and software product evaluation. He is a trained Software Engineering Institute CMM®-Based Appraisal Software Capability Evaluator. He is has served as an evaluator for the National Information Assurance Partnership (NIAP) commercial product evaluation program. In conjunction with the National Institute of Standards and Technology (NIST), he has conducted research and development and practical application in role-based access control (RBAC) architecture and role engineering.

Dr. Coyne participates on several standards development organizations to promote the development and adoption of standards in healthcare information technology. These include Health Level 7 (HL7) Security Technical Committee, ASTM International's Committee E31 on Healthcare Informatics, the OASIS Access Control Technical Committee, the Certification Commission for Healthcare Information Technology (CCHIT) where he serves as cochair of the Security Expert Panel, and the InterNational Committee for Information Technology Standards (INCITS) where he serves as Chair of the Task Group on Role-Based Access Control. The Task Group is developing a national standard for the implementation of RBAC infrastructure. For HL7 he authored a Common Criteria protection profile for CCOW context managers and applications.

For the Veterans Health Administration (VHA) Dr. Coyne has developed guidance for role engineering and participates on the VHA healthcare RBAC task force to identify standard roles and permissions. He has prepared Common Criteria protection profiles on VHA's VistA legacy system, Common Services, VA Smart Card and Biometric Protection, and Health Data Repository.

He authored a paper on role engineering for the first ACM workshop on RBAC in which he coined the term "role engineering" and coauthored a fundamental article on role-based access control models that appeared in *IEEE Computer*.

Dr. Coyne has taught courses at The American University as a professorial lecturer in the engineering of real-time systems. He holds a B.S. in astronomy and an M.A. in the history of science and technology from Case Institute of Technology, an M.A. in linguistics from The American University, and a Ph.D. in theoretical linguistics from Georgetown University.

John M. Davis is the security architect for the Veterans Health Administration (VHA), Office of the CIO, Emerging Health Technologies and Standards Offices. His areas of expertise encompass privilege management infrastructure, role engineering, cryptography, and secure software development.

Mr. Davis is the author of VHA's "Security Blueprint," a vision for VHA's service-oriented security architecture. He leads architectural efforts to improve clinician access to electronic health records through single-sign on technology, emergency access, role-based access control, electronic signature, Federated Identity Management, secure messaging, information rights management, secure software development and enterprise-wide security audit services. He represents and leads VA participation on several Standards Development Organizations and HHS-sponsored groups including ASTM International's Committee E31 on Healthcare Informatics (Cochair Privilege Management Infrastructure Subcommittee), Health Level 7 (HL7) (Cochair Security Technical Committee where he led an effort within VHA and HL7 resulting in the first-ever worldwide standard supporting role-based access control in healthcare), the Certification Commission for Healthcare Information Technology (CCHIT), US TAG to ISO TC215, the American Health Information Community (AHIC), and ANSI INCITS CS1, developing healthcare industry standards for security of electronic health records.

For the VHA and the HL7 organization Mr. Davis defined a role engineering process for use by the VHA RBAC Task Force and HL7 efforts to standardize healthcare permissions. He has developed guidance for role engineering and oversees the efforts of the VHA healthcare RBAC task force to identify standard roles and permissions.

Mr. Davis is a security coauthor of "Person-Centered Health Records" and "Modeling Privilege Management and Access Control," published in the *International Journal of Medical Informatics*. He holds an M.S. in physics from Texas Tech University and an M.S. in electronic engineering from the Naval Postgraduate School.

Index