

Towards an Integrated Approach to Role Engineering

Chris Giblin, Marcel Graf, Günter
Karjoth, Andreas Wespí
IBM Research – Zurich
[cgi|grm|gka|anw]@zurich.ibm.com

Ian Molloy, Jorge Lobo, Seraphin Calo
IBM Research – Watson
[molloyim|jlobo|scalos]@us.ibm.com

ABSTRACT

Although role-based access control has become a preferred method to manage access control, it constitutes a significant effort to develop and maintain a role structure. Role engineering, the process of defining roles and assigning permissions and users to the roles, aims to define an accurate and complete set of roles using a variety of inputs. In this paper, we describe a unified approach to role engineering supporting a combination of different methodologies, and its partial implementation in the IBM Tivoli Role Modeling Assistant, a role engineering platform reflecting the dual importance of top-down and bottom-up data collection and analysis. Data, imported from multiple sources such as LDAP registries, human resource extracts in CSV format as well as from interviews with the organization's users and subject matter experts, can be browsed, filtered, and visualized. Roles can be created and edited manually or generated automatically from mining results.

Categories and Subject Descriptors

D.4.6 [Operating Systems]: Security and Protection—*Access controls*; H.2.8 [Database Applications]: Database Applications—*Data mining*; H.5.2 [Information Interfaces and Presentation]: Graphical user interfaces (GUI)

General Terms

Security, Design, Management

Keywords

RBAC, role engineering, role mining, visualization, compliance

1. INTRODUCTION

Role-based access control (RBAC) is widely used in enterprise security and identity management products. Among its many advantages, RBAC supports identity governance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SafeConfig'10, October 4, 2010, Chicago, Illinois, USA.

Copyright 2010 ACM 978-1-4503-0093-3/10/10 ...\$10.00.

and compliance as follows. First, it reduces the effort required for certification by reducing the number of relations that have to be managed. Second, it makes authorizations understandable to non-technical business users by giving them a name and relating them to one another. This way roles serve as a bridge between business and IT.

However, building an RBAC system is the costliest part of migrating to an RBAC implementation. Any improvement to methodology which can reduce the cost of RBAC system creation will further increase the effectiveness of RBAC and accelerate its adoption.

Role engineering, as introduced in [3], is the process to define roles and assigning permissions to the roles. Being essentially a requirement engineering process, the goal is to define an accurate and complete set of roles. Several methodologies for establishing a set of roles satisfying a variety of criteria have been proposed as well as software tools to assist in carrying out the methodology. However, there is no common agreement on the quality measures for the assessment of the generated role definitions, minimality, interoperability, and stability for example [5].

There are two general approaches to the construction of an RBAC system. In the top-down approach, which we also refer to as *role modeling*, people perform an analysis of business processes, interview line of business owners, and derive roles based on this investigation. While the top-down process produces meaningful and well understandable role structures, it is considered to be slow and human-intensive.

To overcome the drawback of top-down approaches, data mining techniques have been developed to discover roles from assignments between users and permissions extracted from system configurations [10, 17]. Such a bottom-up approach is called *role mining*. Role mining can potentially accelerate RBAC system construction to a great extent since it can automatically propose roles to role management administrators. However, role mining alone has its limitations. Role mining cannot be merely finding frequent itemsets from user-permission associations. A practical role mining approach should be able to justify the meaning of the roles it has discovered.

Moreover, RBAC systems are not static. Once an RBAC system is built and put into use, it must be administered, maintained and re-certified. Over time, an RBAC system is updated to meet the changes on access needs in an organization. Role obsolescence is a challenge that needs to be addressed in evolving RBAC systems.

The IBM Tivoli Role Modeling Assistant (RMOA) is an experimental platform combining both top-down and bottom-

up techniques for role discovery and authoring. By combining both techniques, administrators can leverage the speed advantages of bottom-up role mining with the increased confidence provided by the top-down approaches. RMoA imports role, organizational and identity data from multiple sources. Subsequent data analysis employs interactive attribute filters, descriptive statistics and data mining techniques to identify role discriminators and automatically infer roles. Visualization enhances the exploration of data and enables the detection of patterns during role design. RMoA is a design-time modeling environment, not an operational tool. It ultimately produces role definitions which are then consumed by operational components such as provisioning systems.

The rest of this paper is organized as follows. Section 2 elaborates on the different activities within role engineering and Section 3 provides an overview of the RMoA software architecture including visualization, data mining and analytics on access management. Section 4 discusses related work and conclusions are drawn in Section 5.

2. ROLE ENGINEERING

Possibly the biggest impediment for manually defining and creating roles is the sheer amount of data. Organizations with thousands of users can easily have millions of resources for which access must be controlled. Although access control roles may share some similarity with organizational roles, permissions do not necessarily follow the organizational hierarchy. Permissions and roles may be better organized according to functional responsibilities and business processes. Creation and administration of roles are usually performed by IT administrators familiar with the business processes of their organization. With the exception of small organizations, however, administrators often have only partial knowledge of their organization and must look to the structure of data for insights into appropriate roles. It is in exploring the data structure that an administrator combines top-down and bottom-up techniques to create the roles.

We view the creation of an RBAC system as a process with multiple iterations of the following tasks: Data collection, data exploration, data cleansing and partitioning, role mining, and role definition and reconciliation. This sequence of tasks is a natural but not strict order. The creation of an RBAC system is interactive with substantial data exploration so that many of these tasks are applied at different times. Next, we describe the tasks in more detail.

2.1 Data collection

The data collected for role creation can be classified into (1) access control data and (2) organizational data. These classes of data may be obtained from both system and human sources.

The basic access control data comprises users, permissions and user-permission assignments. Attributes associated with users and permissions may further help in assigning meaning to user-permission assignments. For example, geographic location of a user's office may explain permissions assigned to that user and can be used as a pivot for a role definition. Organizational traits such as reporting structure can sometimes be inferred from user attributes and also exploited. Such information is intrinsic to an IT system. Tools can be built to automatically or semi-automatically

import the data from configuration files, directories and other databases into the role modeling system.

Acquiring information beyond that which is available electronically is a more manual process, obtained through interviews. In an interview, the role engineer queries a person, such as a business process owner or system administrator, for insight into the known or intended users of systems, processes or applications. At a minimum, the role engineer asks three questions: (1) What systems are used in your area? (2) Describe roughly the different job functions in your area. (3) For each job function, list the systems which need to be accessed. Interviews can serve to establish the business context of user-permission assignments. Although coarse-grained, this information can also be used to check the validity of user-to-permission assignments and determine whether data cleansing is needed. Interviews can be conducted through a variety of means such as a face-to-face meeting, a phone conversation, a web-based questionnaire or email. The results of interviews are semi-structured and may contain ambiguities. For example, one interviewee might refer to the "Accounting System" while another interviewee might refer to the same system as "ACS". Such inconsistencies must be resolved later by the role engineer.

2.2 Data exploration

Role engineers explore data at various points in the process, formulate queries returning a statistical summary for example, preferably presented in information-rich visualizations. Statistical techniques like clustering, factor analysis, or matrix decomposition are used to discover user, permission, and attribute clusters which reveal patterns in the organization [6, 12]. Visualization enhances the exploration of data to reveal patterns relevant to role design and should be easily comprehended by both security and non-security professionals.

After collecting data, the role engineer needs to assess the quality of the different sources. Data quality is an important factor in determining the role engineering strategy, as roles derived from bad data have questionable value. It is common to find incomplete or outdated data which may lead to inconsistencies (see data cleansing below).

Data about users, permissions, group membership, and roles are largely categorical or binary as opposed to quantitative and continuous. Frequencies and histograms are natural tools for understanding the distribution of such data. Further, the distribution of null values helps in assessing data completeness, and a role engineer may cross-correlate different data sources (for example, multiple attributes), to ensure the consistency and relevance of the data.

Often the organizational structure of the company, such as organization charts, is not sufficient to place roles reliably into an organizational context. The role engineer needs to discover first a detailed structure from the data. For example the *manager* attribute allows one to reconstruct the reporting hierarchy and users can be grouped accordingly. This grouping can be compared to a grouping by department to understand the management structure of departments. This can further be validated against work location and other geographical data.

Finally, the role engineer must gather enough evidence from the data to justify the creation of a role from the user-permission assignments. This is often gained from compliance with user attributes. Because user-permission assign-

ments are created to support the organization and not made randomly, one should expect much of the data can be compressed into roles that should be supported with evidence from both the business and IT sides.

For example, a set of user-permission assignments where all users with the “surgeon” job title share the “sign surgical report” permission provides both business and IT evidence for creating a *Surgeon* role. Being able to visualize the distribution of user attributes within a given user-permission set and projecting them onto the permissions, or vice versa, is a useful exploration mechanism for defining roles.

2.3 Data cleansing and partitioning

Roles are introduced to compress the user-permission assignment relation so that if n users share m permissions instead of dealing with $n \times m$ user-permission assignments a role is introduced requiring a total of $n + m$ assignments. However, it is not clear that introducing a role will help in the administration when n or m are small. Measures of goodness have been proposed that can strictly order or prune roles based on the number of assignments [2, 10, 11]. Some are parameterized and allow a user to define how important having more or less user-permission, user-role, or role-permission assignments are [10, 11].

Some role mining algorithms are particularly sensitive to characteristics in the user-permission data and may become overfit. For example, it may be difficult to find significant patterns in small sets of user-permission assignments. Small patterns (small n and m) may be exceptional or special cases for which roles might not be required, or roles might need to be justified for organizational reasons¹. These assignments may also be mistakes due to errors in the data. See [12] for further discussion on the two types of noise in access control data. In our experience, we have often encountered user-permission assignment relations having very long tails. For example, we have observed cases where as few as 10% of the permissions are assigned to many users, with the numbers of users assigned per permission dropping very rapidly thereafter.

Exceptions in the data can make the role engineering task more difficult. Our few experiences with real, operational engagements has shown that practitioners frequently use an 80/20 rule (any $\epsilon/100 - \epsilon$ rule may be defined) whereby roughly 80% of the user-permission assignments are covered by roles and the remaining 20% is treated as exceptions. Data cleansing techniques can be used to select the most appropriate 80% of the data for role engineering [12].

Organizational or administrative reasons can also affect how the data is partitioned. For example, an organization may enforce a constraint whereby roles are not allowed to cross divisional or geographic boundaries.

A further reason for partitioning the data relates to scalability. Large datasets in general pose problems when attempting to inspect, visualize or mine data. This partition can be based on a combination of clustering results and attribute based partitions.

Finally, how access control enforcement is performed may also influence the partitioning of data. For example, a role may contain a set of permissions which, at the implementation level, spans more than one system (e.g., some permissions correspond to a database access, while others to Active

¹These roles will probably be very unstable—see the reconciliation task for more on stability.

Directory, and still others to custom applications). While at a high level, these permissions may be grouped together, at an operational level, they are treated as distinct sets of permissions, since no mechanism is in place to provide the enforcement across applications. In this case, the data may best be partitioned along system boundaries.

2.4 Role mining

Role mining is the process of discovering roles and role hierarchies automatically that fit *best* the user-permission assignment and possibly some top-down information using different data mining techniques. In the last few years, several role mining algorithms have been developed. A comparison of many of these algorithms can be found in [11].

The definition of best fit is subjective and depends upon the specifics of an environment. Ideally, the role engineer can define criteria to influence the mining such as the depth of role hierarchies and any predefined roles which should be considered during mining. Other criteria to consider during mining include the number of roles, and the number of permissions or users per role.

The result of mining can be provided in two forms. One form is a full RBAC state: a suggested set of roles, a role hierarchy and possibly a set of user-permission assignments not covered by these roles since they do not fit the criteria. In the other output form, mining finds patterns in user-permission assignments and returns candidate roles one at a time until some portion of the user-permission assignments is covered.

2.5 Role definition and reconciliation

With partitions defined using visual and statistical clustering and with organizational data collected, the creation of roles can start. Some roles may be directly mandated by an organization’s policies or by regulations and therefore easily predefined. During role definition and reconciliation, the role engineer must also take the suggested mined roles and incorporate them as system roles if they are considered appropriate. In this step the administrator may reject roles, modify roles, or merge and split roles. Also, appropriate role names need to be assigned to mined roles². Particular attention must be paid to roles which were manually defined and passed to the role mining as predefined roles. These roles could have been modified, in some cases even ignored, necessitating a human decision.

With the ability to compare roles, roles can be mined according to different criteria to find similarities and differences between multiple mining runs. The same comparison can also be performed against those roles and role hierarchies which have been defined manually. The Jaccard coefficient was used in [16] to develop a mechanism to measure differences between individual roles and RBAC systems. In [12], maximum weighted bipartite matching was proposed which has the advantage of being a real metric given a distance metric between two roles and hence is a more appropriate way of measuring distance.

A good role should be resilient to change; i.e., if users move within the organization or leave the organization or new users join, the roles, defined as a collection of permissions, do not need to change. Thus, we are interested in *stable* roles. During reconciliation, role stability could be

²So far no role mining algorithm returns roles with suggested names.

tested. One possible way to check stability is to partition the user-permission assignment randomly, run role mining in the different partitions and check that roles keep reappearing. Stability can also be determined not by exact match between roles but by low distance using a metric. Methods to measure stability have been defined in [2, 12].

Finally, reconciliation should consider the verification of separation of duty constraints and other operational restrictions.

2.6 Example Use Case

The tool described in this paper is designed to allow the role engineer maximum flexibility to experiment with different cleaning, partitioning, and role mining and role definition tasks. Any step will typically be combined with the data exploration and visualization to better grasp the changes a new set of parameters has on the final RBAC state. Each task can be executed and re-executed in any order until the role engineer is satisfied with the final results with the tool allowing seamless movement between each task. For example, see Figure 1. A role engineer may begin by (1), importing the interview and (2) the existing user-permission access control data before exploring the data. The aggregate data is then normalized into the uniform schema (3), after which it may be cleaned and partitioned (4), before initial role definitions are created (5). The cleaned and partitioned data may also be mined (6), before being reconciled with the manually defined roles (7). When the role engineer is satisfied, the final RBAC configuration is output (8). Because an access control system is always changing, the resulting access system may be reimplemented (9) at a later date to update the RBAC system.

Alternatively a role engineer may decide to initially perform role mining to obtain an initial RBAC state which can be inspected in the exploration and visualization stages. Due to the size of many organizations, cleaning and partitioning may be required. Role engineers will find themselves moving between exploration, cleaning and partitioning, and role mining with different parameters until they are fully satisfied with the results. From there they may move to role definition and reconciliation to manually make desired changes before outputting the final RBAC state. These represent just some of the possible workflow patterns supported by the Role Modeling Assistant, described in the next section.

3. ROLE MODELING ASSISTANT

The IBM Tivoli Role Modeling Assistant (RMOA)³ is an Eclipse Rich Client Platform (RCP) application embedding a Derby database for relational data and the Jena triplestore for RDF data. A network-enabled standalone application, RMOA allows role engineers to collect data in one step and analyze data in a separate step, possibly offline.

Eclipse RCP provides a number of inherent capabilities which are well-suited for a role engineering platform. Its notion of project allows the role engineer to keep multiple role engineering activities organized and separate, providing a natural scoping and partitioning mechanism. The multi-artifact capability of Eclipse encourages the storing of related documents in the project, such as requirements documents and presentations, even if they are not directly ana-

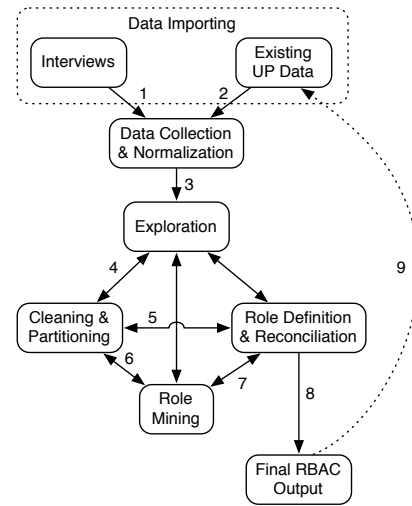


Figure 1: Example workflow pattern.

lyzed by RMOA. Eclipse’s plug-in component model permits RMOA’s role modeling plug-ins to be mixed and matched with other plug-ins important to the role engineer such as XML editors and LDAP browsers.

Additional libraries have been used in developing higher-level functions. Domain objects have been implemented with Jena or generated with the Eclipse Modeling Framework (EMF). The charting library is a subset of the graphics developed in the ManyEyes project [18].

3.1 Architecture

The architecture of RMOA is shown in Figure 2. Key high-level functions appear on the uppermost layer of the figure: importing of data, exporting of role definitions, viewing and visualization of data, editing of roles, constraints, and role mining. These functions operate on domain objects representing users, groups, permissions, systems and roles. The domain layer includes an Object Relational Mapping (ORM) for transforming domain objects to and from data storage.

The database layer employs two storage models: relational and triplestore. Except for roles, the domain objects reside in the relational database to accommodate efficient querying of very large sets of user and permission data. Roles are captured as graphs in RDF and persisted in the triplestore. Triples are well suited for describing role hierarchies and arbitrary attributes associated with roles. Further, the concept of globally unique URIs plays an essential role in versioning and exchanging role definitions between systems.

3.2 Metamodel

Given the diversity of directory schemas, access control models and data formats, RMOA maps external data onto a metamodel of the core concepts of user, group, role and permission. The metamodel, shown in Figure 3, promotes a normalized view of a heterogeneous environment.

The user model is derived from the inetOrgPerson schema specified in RFC 2798. As such, User comprises twenty attributes including *cn*, *employeeNumber*, *mail*, *o*, *ou*, etc. For performance and simplicity, multiple-value attributes are mapped onto single-value attributes. User records can be progressively updated with additional data sources, provided a key, such as an email address or employee number,

³Available as open beta at <https://www14.software.ibm.com/iwm/web/cc/earlyprograms/tivoli/tivrole/>

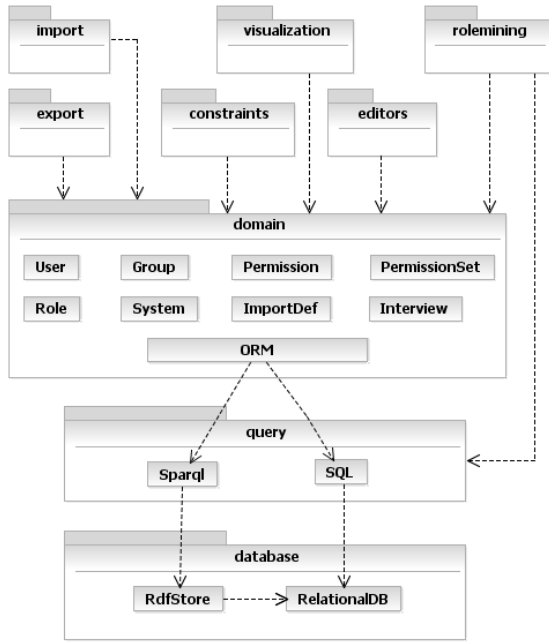


Figure 2: Logical architecture.

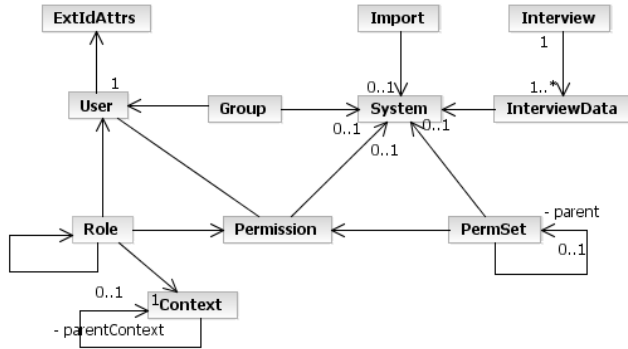


Figure 3: RMoA metamodel.

can be used as a correlator. Likewise, correlator attributes are used to link group memberships with users. The user schema can be extended with arbitrary name/value pairs, however with a performance penalty as the number of these extended attributes increases.

The permission model is a technology-neutral abstraction aligned with the RBAC reference model. That is, a permission is a tuple consisting of an object and an operation performed on that object. Additional fields are included for a description and a display name. In the interest of simplicity, the permission model does not support object hierarchy semantics such as containment and traversal nor does it support rules such as temporal restrictions. While sufficient to support role modeling, it is not an operational permission. Nevertheless, the model accommodates coarse- and fine-grained permissions of varying abstraction levels. Permissions are imported into the Role Modeling Assistant in CSV file format.

The interview model is a flexible schema for collecting

human knowledge about access control permissions. The interview schema contains metadata about the interview itself (interviewer, interviewee, a timestamp) and any number of access descriptions. Access descriptions consist of information about the permitted users, the system(s) involved, an operation, a target object (i.e., the resource to be accessed) and an optional free text description.

General-purpose entities in the metamodel include permission sets and import definitions. *PermissionSets* group related permissions into a single, high-level permission, thereby providing the role engineer a means to manage and reason more abstractly about permissions. Permission sets may contain permissions from multiple systems and may be arranged in a simple tree hierarchy. The *Import* entity in Figure 3 relates an individual data record to the import event which brought it into the RMoA system. Thus every record of data can be traced to the source whence it came.

3.3 User Interface

As an Eclipse RCP application, the user interface is built on the concept of a perspective. A perspective is a configuration of related information displays, called views and editors. RMoA exploits perspectives to provide separate working environments tailored to the data importing and role modeling activities.

The data perspective addresses the importing and management of data. Import connections to data sources such as LDAP, Active Directory and CSV files are configured here. Once defined, the connections can be launched to retrieve external data, which is stored internally in terms of the RMoA metamodel. Optional mappings between source and target attributes can be defined and applied during import. Imported data can be deleted, re-imported and browsed. This perspective also includes views for displaying import connections, import history and statistics, data mappings, and a view for reporting constraint violations.

The RMoA role modeling perspective, shown in Figure 4, provides tools supporting the creation and editing of roles and their hierarchies, as well as visualization capabilities. This perspective includes the attribute comparison view, a large matrix for browsing and filtering user, group and permission data. User names are enumerated along the top horizontal axis and user attributes on the left vertical axis. Each cell is the intersection of a user and an attribute. The cell contains the value of that attribute for the given user. The vertical axis also includes all imported group and permission names.

To select and work with a subset of users, entry fields for filters appear alongside the attributes. For user attributes, the filter entered is a regular expression. For groups, a filter selects group membership or non-membership. The permission filter selects assigned or non-assigned users. For example, to select all members of the accounting department who are also members of the payment group, the *deptNumber* filter is set to “accounting” and the payment group’s filter set to “Members”. The matrix is automatically updated to display only those users satisfying the selection criteria.

Not visible in Figure 4, a role hierarchy navigator is available for viewing and working with the role structure. This navigator employs a scoping mechanism called a folder. Folders, which can be nested, enable the development of independent role hierarchies and are useful for organizing roles within a context. The navigator can be toggled between

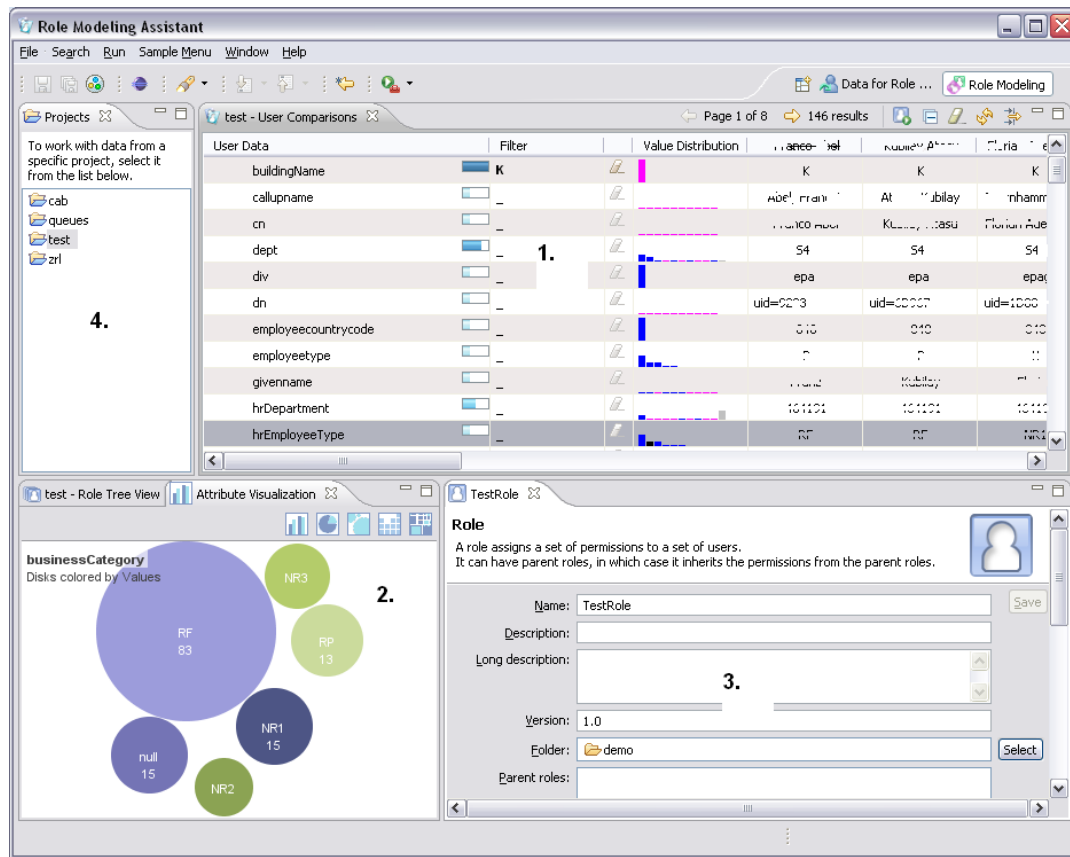


Figure 4: RMoA role modeling perspective: 1) Attribute comparison view. 2) Attribute visualization. 3) Role editor. 4) Project selector.

two views. A folder view displays the hierarchical structure of folders with roles enumerated in a flat fashion. The second view shows the same data but organized according to the role hierarchy. To represent multiple inheritance in the navigator's tree widget, annotation and node replication are used.

3.4 Visualization

Visualization helps to answer questions about data which are essential to the role engineer during design. The role engineer needs to discern attributes which are discriminators, that is, attributes whose values can potentially be used in defining a role. As the role engineer works with filters, further insight is needed into how an attribute distribution corresponds to the current filter selection. Equally important, uninteresting attributes should be quickly identifiable in order to suppress them from the current working view.

To address these requirements, three types of visualization are employed in RMoA's role modeling perspective: histograms, charts and a score indicator.

Small, sparkline-inspired color-coded histograms appear in the attribute comparison view to quickly convey properties of each attribute. Foremost, the histograms summarize the relative frequency distribution for attribute values under the current filter. Frequency bars are displayed in descending order, from left to right, such that the most frequent values appear on the left. To keep the histogram compact, a maximum of ten bars is displayed. Long tail values, if

present, are collected into a final, gray-colored bar. The frequency of null values is presented as a black bar. Finally, the histogram also communicates whether the distribution is wholly contained in the current filter selection. In the color-coding scheme of these histograms, magenta bars denote values which are wholly contained in the current filter set; blue bars indicate that users with the same value exist beyond the current filter setting.

The interpretation of histograms is now illustrated with examples. In Figure 4, the histogram columns are uniform and of minimal height for the attributes, *cn* and *dn*. As these attributes are typically unique for each user, the frequency of any value is one, as expected. At the other extreme, the histogram for the *div* attribute has one bar of maximum height, indicating the value is the same for all users. The attributes *cn*, *dn*, and *div* are therefore uninteresting as discriminators and can be suppressed from view. The histograms for attributes *employeeType* and *dept*, however, reflect more diverse distributions and are candidates for discriminators. The histogram for *hrDepartment* demonstrates the use of the final gray bar, collecting the lowest frequencies into the final bucket. The black bar in the histogram for *hrEmployeeType* indicates the presence of null values.

A score meter appears next to each attribute indicating how attributes fit the current filter selection. A categorical attribute (such as department or work location) partitions the set of users into disjoint subsets in the following way: users that share the same attribute value are in the same

subset. We say that the current filter selection is aligned with an attribute if the boundary of the selection coincides with the subset boundaries. The better the alignment, the higher the score. A high score indicates that the attribute provides a simple business explanation for the current filter selection.

Charts provide detailed, alternate visualizations of attribute distributions with some interactive capabilities. When the attribute visualization view is active, as is the case in the lower left of Figure 4, a chart is displayed whenever an attribute name is selected in the attribute comparison view.

Supported charts include bar, pie, size-proportional bubble, stacked histogram and tree map. For examples of the chart types, see [18]. Bar charts provide a zoom-in facility for the sparkline histograms, showing exact frequency counts and displaying the tails contained in the histogram’s gray bars. Pie charts display relative frequencies in terms of percentages. Tree maps are sorted, size-proportional charts with a zoom-in function. Bubble charts, present frequencies as size-proportional circles, enabling quick identification of large and small counts. The bubble charts also provide a bivariate view, displaying the distribution of one variable relative to another.

3.5 Role Mining

The role mining module in RMoA builds an initial role structure applying formal concept analysis to user-permission assignments. This structure is optimized according to input parameters determining the role hierarchy depth, number of user-role assignments, number of role-permission assignments, and number of direct user-permission assignments. These mining parameters, combined in a quality measure termed weighted structural complexity, and the mining algorithm are discussed in detail in [10].

Prior to launching, the mining scope is set either to the entire project or a subset using filters as described earlier. Once started, a pre-report is computed summarizing the input data and presenting the parameters for role structure optimization. Figure 5 shows the pre-report with sliders to configure the mining parameters. When mining has completed, a summary report is presented with candidate roles being inserted into the role hierarchy view with generated names such as *Role2*, *Role3*, etc. Developing techniques for role names reflective of role semantics is the topic of future work.

Initial experience with RMoA suggests that course-grained permissions tend to be gathered during the data collection phase. Often, the role engineer obtains information only on accounts and group memberships. Group membership often indirectly represents a set of permissions. For example, membership in a group in a registry may be used by access control lists (ACL) defined in an access control system in another system. Identity and access management provisioning systems often use groups as the unit of granularity for provisioning. To accommodate these use cases, the mining module is able to treat group memberships as permissions.

3.6 Interviews

Interview data is imported from CSV files into RMoA’s internal data store. The role engineer has two choices during import: to import interview data directly as permissions or to import simply as raw interviews. As raw interviews, the data can be consolidated and edited to refine and remove

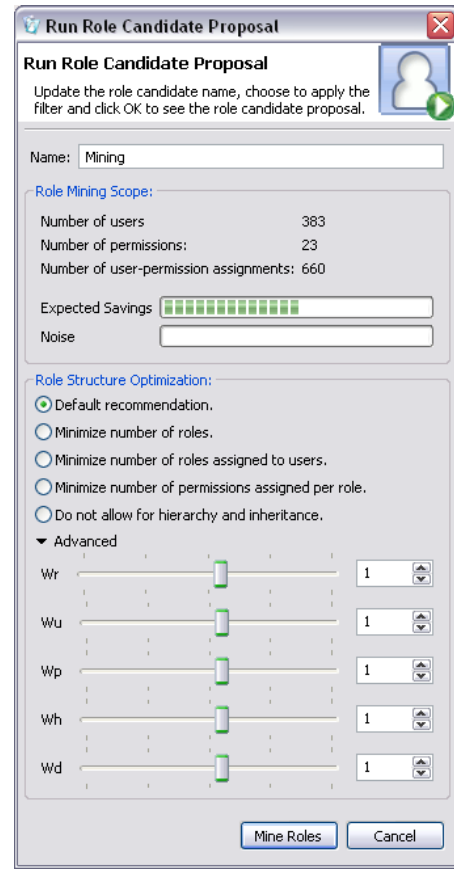


Figure 5: Role mining pre-report and optimization parameters.

| Role | Description |
|----------------|----------------------------------|
| Mining:Role 0 | ZRLall,zrl-cloud-forum |
| Mining:Role 9 | |
| Mining:Role 1 | ZRLall,zrl-svn-privacy |
| Mining:Role 14 | zrl-cryptowiki |
| Mining:Role 17 | zrl_security |
| Mining:Role 18 | zrlcrypto |
| Mining:Role 16 | zrl_role_management,zrl_security |

Figure 6: Candidate roles generated during mining.

ambiguities. Individual elements of the interviews then can be selectively promoted to permissions or roles, thereby being merged into the modeling base while maintaining traceability to the original interview.

4. RELATED WORK

Many approaches for top-down as well as bottom-up role engineering have been described in the literature. Recently, the focus has been on combining both approaches as in hybrid role mining [7] that additionally takes business information such as job function, department or location as input. Role mining algorithms supporting this functionality are described in [1, 6, 10]. In our work, we go a step further by espousing a flexible approach of combining top-down and bottom-up role engineering and supporting various integration patterns.

Graphical tool support, mainly for role structure navigation, view and manipulation, are described in [4, 13]. The

provided operations support management but not role design. xoRET [14] is a graphical software tool supporting a scenario-driven role-engineering process [15]. It facilitates the specification and inspection of trace relations to ease change management and to propose a (preliminary) policy rule set.

There are also tools for the analysis of Administrative RBAC policies, which support analysis of properties such as reachability, availability, containment, and information flow [8, 9]. These tools are complementary to the functions presented in this paper. Finally, there are a number of commercial products available which provide various forms of support for role engineering.

5. CONCLUSIONS AND FUTURE WORK

During the development of RMoA we were consulting and validating the different features with practitioners. These interaction showed that role engineers will interleave top-down and bottom-up techniques during the definition of roles. The IBM Tivoli Role Modeling Assistant provides support for the definition and modeling of enterprise roles. It aids role engineers in the process of collecting user and entitlement data to build roles from bottom-up, while also streamlining collaboration between line of business and IT stakeholders with top-down analysis. This functionality provides a foundation for understanding which privileges users have and which privileges users should have.

RMoA allows importing not only of user and entitlement data, but also data gathered during interviews. It provides data exploration and analysis capabilities including sophisticated role mining techniques. Emphasis has been placed on strong visualization capabilities to support the selection of data relevant to role definition.

RMoA emphasizes flexibility in supporting diverse data collection, exploration, and analysis workflow patterns. Role modeling and mining is a process requiring multiple iterations to arrive at suitable role definitions. The strong integration of the various role engineering tasks, the flexibility in choosing subsets of data to be analyzed, and the integration of subset analysis resulting in coherent roles are key goals. We are currently working with system administrators and role engineers to evaluate the effectiveness of the tools available in RMoA to move between top-down and bottom-up role engineering using the role mining and visualization and exploration tools.

We plan to extend this work by deploying the multidimensional data model capabilities of Online Analytical Processing (OLAP) in the top-down analysis, exploiting advanced data analytics. Other future work includes extending role mining and data cleansing (possibly via a plug-in architecture) to employ machine learning techniques such as those described in [6, 12]. As a general extension, we pursue the improvement of role engineering methodologies and their tight integration with tools. Ultimately, the goal is to provide the role engineer with an integrated framework enabling the development of high quality role models in an easy and straightforward manner.

6. ACKNOWLEDGMENTS

We acknowledge and thank IBM Tivoli, in particular Todd Jordan and Chrystian Plachco, for the excellent collaboration and the support of this project.

7. REFERENCES

- [1] A. Colantonio, R. Di Pietro, A. Ocello, and N.V. Verde. A formal framework to elicit roles with business meaning in RBAC systems. In *SACMAT '09*, 2009.
- [2] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. Verde. Mining stable roles in RBAC. In *24th Int. Information Security Conference, SEC '09*, IFIP 297, pages 259–269. Springer, 2009.
- [3] E.J. Coyne. Role engineering. In *RBAC '95*. ACM, 1996.
- [4] D.F. Ferraiolo, R. Chandramouli, G.-J. Ahn, and S.I. Gavrila. The role control center: features and case studies. In *SACMAT '03*, pages 12–20. ACM, 2003.
- [5] M. Frank, J.M. Buhmann, and D. Basin. On the definition of role mining. In *SACMAT '10*, pages 35–44, 2010.
- [6] M. Frank, A. P. Streich, D.A. Basin, and J.M. Buhmann. A probabilistic approach to hybrid role mining. In *CCS '09*, pages 101–111. ACM, 2009.
- [7] L. Fuchs and G. Pernul. HyDRo - hybrid development of roles. In *4th Int. Conf. on Information Systems Security (ICISS 2008)*, LNCS 5352, pages 287–302. Springer, 2008.
- [8] M.I. Gofman, R. Luo, A.C. Solomon, Y. Zhang, P. Yang, and S.D. Stoller. RBAC-PAT: A policy analysis tool for role based access control. In *15th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, LNCS 5505, pages 46–49. Springer, 2009.
- [9] K. Jayaraman, M. C. Rinard, M. Tripunitara, V. Ganesh, and S. Chapin. Automatic error finding in access-control policies. MIT-CSAIL-TR-2010-022, MIT, 2010.
- [10] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S.B. Calo, and J. Lobo. Mining roles with semantic meanings. In *SACMAT '08*. ACM, 2008, pages 21–30.
- [11] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang, and J. Lobo. Evaluating role mining algorithms. In *SACMAT '09*, pages 95–104. ACM, 2009.
- [12] I. Molloy, N. Li, Y. A. Qi, J. Lobo, and L. Dickens. Mining roles with noisy data. In *SACMAT '10*, pages 45–54. ACM, 2010.
- [13] S. L. Osborn, Y. Han, and J. Liu. A methodology for managing roles in legacy systems. In *SACMAT '03*, pages 33–40. ACM, 2003.
- [14] M. Strembeck. A role engineering tool for role-based access control. In *3rd Symp. on Requirements Engineering for Information Security (SREIS)*, 2005.
- [15] M. Strembeck. Scenario-driven role engineering. *IEEE Security & Privacy*, 8(1):28–35, 2010.
- [16] J. Vaidya, V. Atluri, Q. Guo, and N. R. Adam. Migrating to optimal RBAC with minimal perturbation. In *SACMAT '08*, pages 11–20.
- [17] J. Vaidya, V. Atluri, and J. Warner. RoleMiner: mining roles using subset enumeration. In *CCS '06*, pages 144–153. ACM, 2006.
- [18] M. Wattenberg, J. Kriss, M. McKeon, F.B. Viegas, and F.V. Ham. ManyEyes: a site for visualization at Internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007.