

An Architectural Approach to RBAC Policy Management Framework in a SOA driven Enterprise

M.K.Raghavendra¹

¹ Merit Systems Private Limited, 55/c, 42/1, Nandi Mansion, 40th Cross,
Jayanagar 8th Block, Bangalore, India, 91-80-26542726
ragh@meritsystems.com

Abstract. Access Control is one of the key components in the security and compliance strategy in any business. Organizations have adopted Role Based Access Control (RBAC) approach to protect business critical resources such as digital assets, computing and network resources, databases, web sites and so on. Enterprises however, are faced with challenges of maintaining disparate and inflexible RBAC implementations coupled with the inherent challenges of dealing with ad-hoc organization structures, dynamic work allocations, multiple or overlapping role allocations, diverse resource types and so on. Evolving standards such as reference RBAC model from NIST [1] and XACML [2] from OASIS address certain aspects of the overall problem. This paper defines an architectural approach for RBAC implementation that attempts to address these challenges leveraging contemporary standards and SOA trends in enterprises.

Keywords: RBAC, Role, Access Control, NIST, XACML, Enterprise, SOA, Process

Introduction

The security and regulatory compliance has taken a centre stage in every business across the globe as evident from the compliance trends such as BASEL II, SOX to name a few. Access Control is one of the key components in the security and compliance strategy in any business. A Role Based Access Control technique is well suited and has gained acceptance in the industry. In the current scenario, IT vendors support proprietary RBAC implementation using variety of techniques, technologies and tools in their applications largely because of lack of a single agreed standard or an approach. In addition the RBAC implementations are tightly coupled with the application code with little or no control to adopt it to changing business needs. All this has resulted in widely varying proprietary access control methods making it very unwieldy to implement and administer the RBAC policies. There are additional challenges for implementing RBAC in an enterprise including ad-hoc organization structures, dynamic work allocations, multiple or overlapping role allocations, diverse resource types and so on.

This paper defines an architectural approach for RBAC implementation that attempts to address these challenges leveraging the SOA trends in enterprises and evolving standards such as RBAC [1] and XACML [2]. The proposed architecture is further enriched by insightful experience gained during the development of BPM platform as a part of internal product development initiative.

1.1 The Premise

Fundamentally, enterprise operations involve people, processes and resources deployed to realize a stated set of goals. In a SOA driven enterprise business processes are realized by orchestrating across several business services modeled as SOA styled software services with well defined contracts or interfaces. A contract or an interface exposes a set of operations with business semantics. People with commensurate roles interact with these operations to perform their job roles. A set of resources are consumed and produced while executing these operations. These resources are protected from unauthorized access by RBAC policies.

Services, as the first class candidates in a SOA driven enterprise provide an operational context for designing, deploying and administering RBAC policies more effectively. The diagram bellows shows a conceptual view of RBAC in a SOA driven enterprise.

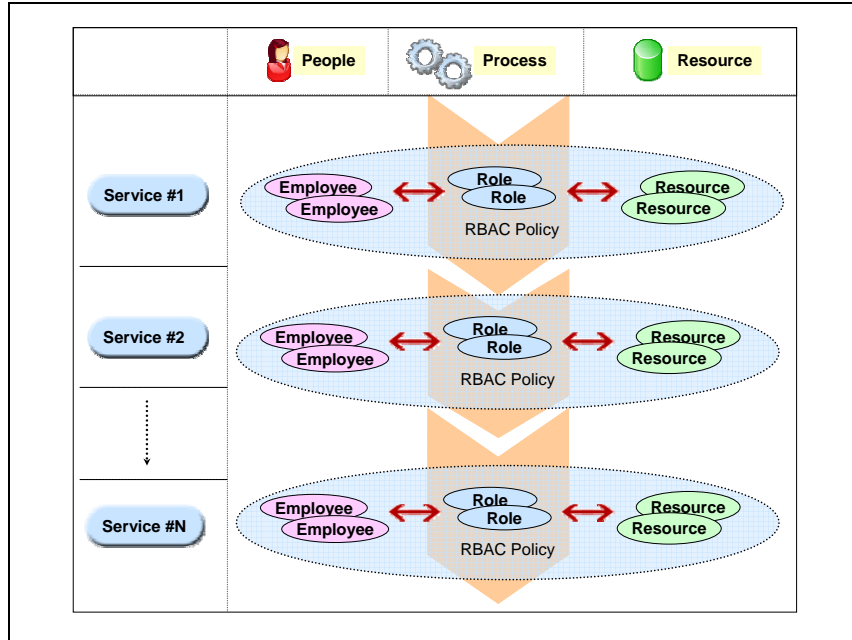


Fig 1 - A Conceptual View of an Enterprise RBAC

A simple illustrative scenario is described here to reinforce the basic idea.

Consider an inventory service implemented as a web service (WSDL) as per OSS Inventory API specification [4] for a telecom operator. The service exposes inventory operations to manage product, service and resource inventories. One of the critical functions is to create and maintain the product, service and resource specifications in the inventory. Clearly the service contract that defines these operations provides a context for identifying the roles that need to perform these operations and the entities that are accessed by these operations which can be used to define a RBAC policy. In this case a RBAC policy can be defined that provides permissions for hypothetical roles such as product manager, service manager and resource manager to create and manage product, service and resource specification documents respectively. The inventory service enforces its RBAC policy when users attempt to access these resources. In general, a SOA styled service can be considered as an independent domain or a logical boundary for designing and enforcing RBAC policies.

Architectural Approach

This section details an architectural approach to RBAC policy management framework based on the premise just established. The architecture is described by two

types of modeling artifacts namely RBAC entity model and RBAC system model. Both the models extend the general abstractions identified in the reference model of NIST [1] and XACML of OASIS [2] into an overall enterprise context that essentially serves as the foundation for an enterprise-class RBAC system. The emphasis here is to demonstrate how these standards can be leveraged in the overall architecture rather than specify a standards compliant architecture.

RBAC Entity Model

This is an overall entity model with the basic RBAC reference model of NIST [1] gracefully extended into larger “enterprise context”. It is represented using UML notations.

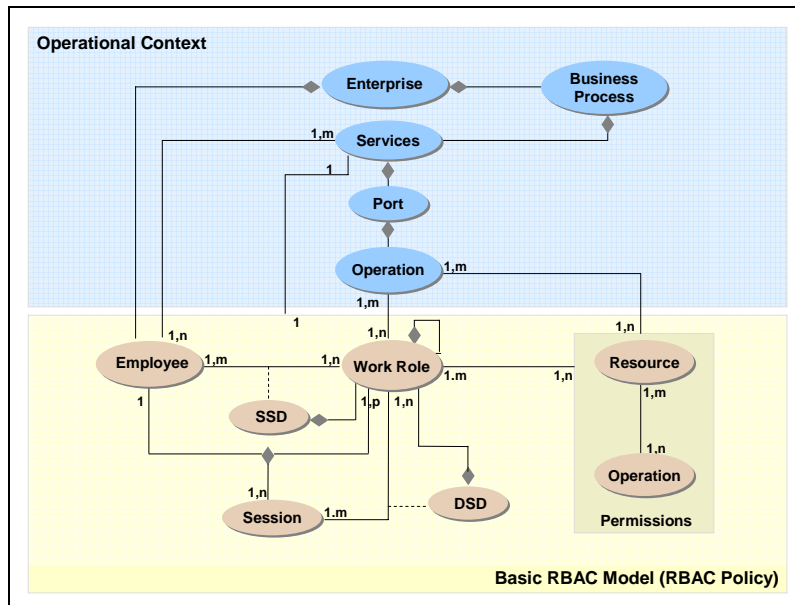


Fig 2 - An Extended RBAC Entity Model

Operational context is characterized by a set of entities namely business process, services, ports and operations. WSDL is used as the reference model to represent the services.

The basic RBAC model is represented by a relationship among the employee, role, resource and permission entities as per the reference RBAC model of NIST [1] represented in UML.

Operational Context Model

The Enterprise entity is a logical representation of the business enterprise within the RBAC system. An enterprise is an aggregation of employees and business processes. A business process, in a SOA driven enterprise can be visualized as a composite application orchestrating among several business services. Each service comprises of a set of ports where each port supports a related set of business operations. Service operations provide the necessary context for identifying resources consumed or produced by those operations. Naturally, they serve as good foundation for designing and maintaining organization-wide RBAC policies more effectively.

Additional motivation comes from various standardization initiatives in the industry. For example OMG's domain services initiative [3] specifies abstract interfaces for services relevant to specific industry domains. CAD service interface for manufacturing domain, Clinical Observation Access Services for health care domain, Access subscription service interface for telecom domain are few examples for the kind of business services being standardized. Similarly, TM Forum's OSS/J initiative specifies interfaces (APIs) [4] for several services relevant to the Telecommunication business such as order management, Inventory, billing mediation, service activation, trouble ticket and so on. They help in establishing common understanding of the structure and semantics of business services and operations and thus serve as a sound basis to RBAC approach.

RBAC policy can be defined for each service to protect the service, its ports, operations and resources required for the operation from unauthorized access. In an enterprise where business processes are formally modeled, the associated business services, ports and operations can be automatically discovered from those models and used as the basis to identify roles, resources and permissions required for these roles to access these resources. This is more straight forward if the process models are represented using standards based notations such as WS-BPEL [5] that are built around WEB services standards, or even BPMN, Role activity diagrams, UML activity diagrams and so on. An administrative application can help the business analyst to define and maintain RBAC policies more effectively using these models. In any case, as a first step in the RBAC design process, it is necessary for a business analyst to establish the operational context in some way and uses it as the basis to design and maintain RBAC policies. This facilitates a formal and disciplined approach to RBAC implementation. In a more advanced scenario the business process and service modeling tools can automatically create the roles, resources and the RBAC policies right at the time of modeling.

A Resource is any entity such as a service, port, operation or any resource required for those operations such as document, application, computer, file, database, software license or even physical entities such as a conference room, projector, telephone and so on that require to be protected from unauthorized access.

Within this model, a resource entity is a metadata object or a descriptor of the actual resource that has to be protected from unauthorized access. It does not control

physical access to the resources it represents. The software services or humans accessing a resource need to explicitly check with the policy management framework if the requesting role has the necessary permissions on a resource being accessed prior to the actual access.

The resource entity carries a unique identifier among other attributes that provide sufficient information about the resource and helps in managing the resources within the RBAC system. The exact set of attributes of a resource entity depends on the type of resource they represent. A typical implementation would involve developing generic resource classes that can be extended with additional attributes and behavior.

An operational context is represented by the static relationships among these entities namely the enterprise, people, process, service, port, operations and resource. This becomes the basis for defining the basic RBAC model.

Basic RBAC Model

An employee with commensurate skills is designated with one or more work roles to execute various operations. One or more employees may be designated with the same work role. There can be a hierarchy of work roles. The work roles here are different from the organizational designations given to the employees. The designations are used in all business communications where as service specific roles are used to define RBAC policies. For example, in one organization the GM Finance may play the role of approver in the budget process of a business unit and also as one of the approvers in a procurement process. But an employee with the same designation in a different organization may not perform the same set of operations. The separation of official designations from work specific role is crucial to a RBAC implementation.

An employee role association is subjected to a Static Separation of Duty (SSD) constraint represented as a role set. No employee can be associated with more than a specified number of roles from the SSD role set.

A session represents an active association of a user (employee) and corresponding roles with a runtime RBAC system. This association is subjected to a Dynamic Separation of Duty (DSD) constraint represented as a role set. No user session can have more than a specified number of active roles from the DSD role set.

A role is assigned a set of permissions required to execute the operations. Permission is an approval to perform one or more operations on one or more resources. Operations are specific to the type of resource. For example if the resource is a document then the possible generic operations are read, write, modify and delete. However in a different context, it may make sense to support additional meaningful operations such as authorize, approve, review and so on and accordingly define additional set of permissions.

RBAC System Model

The system model is described with respect to a set of components and their interactions as a simplified subset of actors and their interactions as describe in XACML model [2]. The service domain component conceptually maps to a Policy Decision point (PDP), the Administration application maps to Policy Administration Point (PAP) and the various enterprise applications and services map to access requester entity in the XACML model. The RBAC policy is the reference RBAC model of NIST represented as a XACML profile. The other entities have been suppressed for brevity sake. The overall functionality of the system is described with respect to interactions between various components. The implementation possibilities based on relevant standards and technologies are briefly described.

At the heart of the runtime RBAC system are shown a set of service domains corresponding to a sample set of business services such as inventory services, general ledger services and so on. Each service domain comprises of a set of services and data repositories.

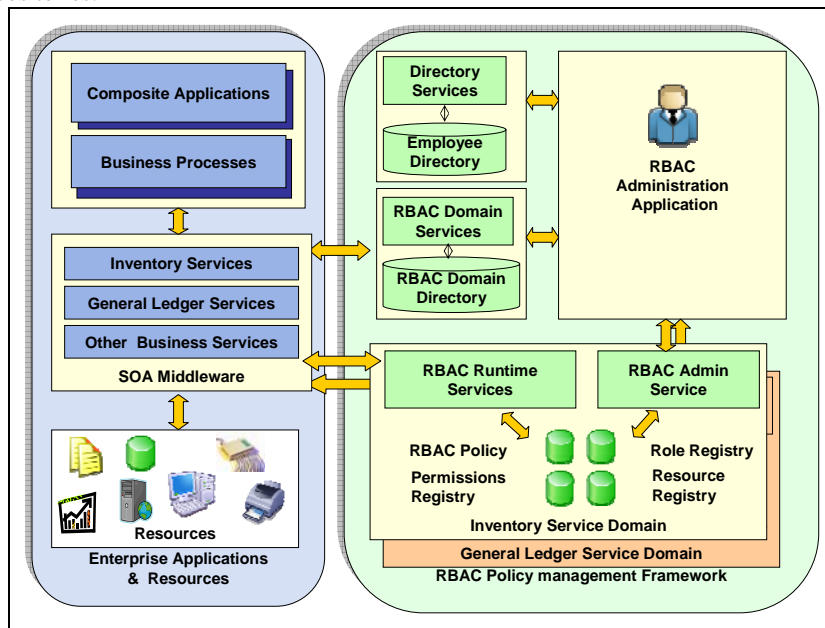


Fig 3 - RBAC System Model

The RBAC Admin service expose functions to create and manage roles, resources, permissions, static and dynamic separation of duty constraints, user role assignments, resource permission assignments and the RBAC policies. A centralized RBAC Administrative GUI application enables an administrator to create and manage these

entities for each service domain. The GUI application in turn interacts with the admin services to create and manage these entities in the respective repositories.

The RBAC runtime services expose functions to perform RBAC policy checks at runtime. The SOA middleware or the business services invoke the runtime services with user, role, resource id, and the operation as inputs. The runtime services check the RBAC policy if the requesting role has the permission to perform the requested operations and accordingly return a grant/deny status. The calling entity accordingly completes or denies the requested operation and returns the corresponding status to the end user.

From a standards perspective, the RBAC Admin and runtime service interfaces can be modeled around NIST functional specification. The roles, resource and permission registry are a collection of roles, resources and permissions specific to the service domain. The RBAC policy repository includes SSD and DSD policies represented as set of roles across service domains. These policies are shared across domains. These databases can be designed around the basic RBAC model implemented as XACML profile for better portability and interoperability.

The RBAC domain service is a centralized service. It exposes functions to create and manage multiple domains in the RBAC system. The Domain directory is a persistent repository of service domain definitions. The LDAP is well suited for domain services implementation. The domain definition includes name of the domain, runtime reference to the admin services, runtime services and databases of that domain. If RBAC services are implemented as web services, the corresponding port addresses serve as references for runtime access. Similarly the database references can be stored as URLs. This provides great amount of flexibility for distributed deployment of domain services. For example, each service owner can deploy the RBAC domain services and databases in separate set of servers with corresponding reference configured in the domain directory. This approach facilitates seamless integration of disparate RBAC implementations for each business service in the enterprise and also a set of RBAC service instances can be shared by multiple service domains for better performance and scalability.

The RBAC administrator creates the domains through the administration application prior to defining the RBAC policies. The application in turn interacts with the domain services to register the domains in the domain directory. At runtime, both the administrative application and the SOA middleware or business services perform a domain lookup through the domain services. The administrative applications use the domain lookup to let the administrator select a domain for defining and managing the RBAC policies. The application retrieves the corresponding reference to the admin services and interacts with it for creating domain specific roles, resources, permissions and policies in local repositories. Similarly, the SOA middleware or business services use the domain lookup to retrieve the reference to the runtime service for the concerned domain and interact with the same to check the permissions (RBAC policy) to perform user requested operation on a resource.

The directory service is a centralized service for managing the employee directory. This is shown as a part of RBAC system in the diagram. However, this is a part of the common IT infrastructure in many enterprises typically implemented using LDAP. Each employee record in the directory contains information such as Employee id, name, department, role (job designation) etc. The administration application provides access to the employee list through this service. An administrator looks up the employee list and maps each employee to one or more roles in one or more service domains as a part of RBAC policy design process.

A Methodical Implementation Approach

A methodical approach to implementing RBAC in an enterprise is briefly described below with reference to various components and their interactions as described in the system model. A more formal RBAC implementation methodology can be evolved out of this at a future date.

1. Study and understand the Enterprise operations – A business analyst studies and understand the overall enterprise operations in terms of various business processes, composite applications and the associated business services.
2. Establish the operational context – The business analyst identifies and establishes the context in terms of service domains, ports, operations supporting roles and resources. The business analyst through an admin application creates the service domains, and corresponding roles, resources, permission sets, SSD and DSD policies in respective registries in each service domain.
3. Define RBAC Policies – The business analyst through the admin application creates the RBAC policies for each domain. The employees are associated with suitable work roles, roles are assigned to appropriate permissions.
4. Configure and deploy – The administrator configures and deploys the work domain services and databases. The business services interact with the RBAC system via the RBAC runtime service interfaces to enforce the RBAC policies.
5. Maintain RBAC policies –An administrator performs various activities on a day to day basis to maintain the policies up to date. This involves activities such as create/modify/delete service domains, roles, resources, permissions, user role assignments, role permission assignments (RBAC policy) in each service domain. Typical scenarios that affect the policies are changes in organization structure, processes & services, roles reallocations or arbitrary work assignments and so on.

Highlights

1. Business centric approach – Using SOA styled business services as the operational context for RBAC design and implementation enables a business analyst to participate more effectively in RBAC life cycle management.
2. Simplified implementation – Ability to define RBAC policy at a business service level makes enterprise wide RBAC implementation simple and more effective. It

also facilitates a methodical and phased approach to RBAC implementation in an enterprise.

3. Adaptability to dynamics of business - The RBAC model execute within the service context. The changes in context in terms of changes in organization structure, processes, role reallocations, dynamic work allocations etc can be gracefully handled with minimal or no impact to the overall system.
4. Scalability & performance – Ability to define RBAC policy at business service level makes it scalable across services.
5. Ease of administration – The RBAC implementation can be administered centrally with multiple service domains across the enterprise.
6. Portability and interoperability – Standards based approach enhances portability and interoperability. Alternate set of standards and technologies can be used to realize a portable and interoperable implementation. For example, choice of web services for core RABC services conforming to the NIST RBAC functional specification, Directory and domain services implemented as web services conforming to LDAP specification and the RBAC data models conforming to NIST reference model or XACML from OASIS can lead to a highly portable and interoperable implementation.
7. Consistent approach- Standards based data models & abstract service interfaces, and a single consistent way of representing heterogeneous resources in the enterprise enable a uniform approach to variety of RBAC needs across the enterprise.

Conclusion

The tough challenges that exist in an enterprise-class RBAC implementation and the technology constraints in providing a perfect solution are fully acknowledged here. Part of the challenge has something to do with unstructured or ad-hoc style of work practices in most businesses. On one side the businesses are moving towards a process and SOA centric approach that enables them to conduct business in a more structured manner as evident from the industry trends. On the other side technology community is striving to address the architectural challenges concerning RBAC. Evolving standards such as RBAC Reference model from NIST [1] and XACML from OASIS [2] play a significant role in this. The proposed architecture has evolved around these trends and provides greater flexibility in handling dynamic access control needs of an enterprise. Arguably this is the most pragmatic approach to RBAC implementation in a business enterprise.

Future Work

RBAC system, similar to single sign-on and LDAP systems, qualifies to be a part of the common enterprise infrastructure. Business process and SOA based solution vendors can design their systems around standards based public RBAC APIs similar to the way applications use LDAP interfaces for directories. Vendors can package RBAC models (say XACML document) along with the process and service models

(WSDL). Installation and configuration applications can automatically setup and configure roles, resources and RBAC policies for each service in RBAC policy management framework. The business processes and services platforms check for RBAC policies prior to performing the requested service operation. This line of evolution can bring in the much needed paradigm shift in approach to the enterprise RBAC. Towards realizing this larger objective following activities are identified as the potential areas of work in near future.

1. Define RBAC APIs based on WSDL conforming to NIST functional specifications
2. Define a comprehensive data model based on NIST RBAC reference model as a profile of XACML
3. Build a prototype implementation of RBAC system based on the proposed architecture and demonstrate the high level service development and deployment usecases based on WSDL standards.

References

1. Proposed NIST Standard For Role-Based Access Control by David F. Ferraiolo, Ravi Sandhu, Serban Gavrilă, D.Richard Kuhn, Ramaswamy Chandramouli (ACM Transaction on Information Security. Vol 4, No 3, August 2001)
2. eXtensible Access Control Markup Language XACML Version 2.0, OASIS standard, Feb 2005. (http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf)
3. OMG Domain Specifications – (http://www.omg.org/technology/documents/spec_summary.htm)
4. OSS/J APIs – SOA Enablement View
<http://www.tmforum.org/browse.aspx?catid=4492>
5. WS-BPEL Specifications
<http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>