

Deliverable AP 5.2

ADMIN: Policy Administration

Requirements for Administration of Security Policies

03.08.2007

Ute Faltin, Fraunhofer SIT
Ute Gerlach, Parks Informatik GmbH
Stefan Kowski, Parks Informatik GmbH
Christian Liesegang, SAP
Ruben Wolf, Fraunhofer SIT

ORKA is funded by the German Ministry of Education and Research (BMBF) as part of its Software Engineering 2006 programme.

© 2007 ORKA Consortium



Federal Ministry
of Education
and Research

Internal document information:

\$Id: del-ap5.2.tex 564 2007-08-07 14:03:29Z gerlach \$

Contents

- 1 Introduction** **4**

- 2 Requirement Analysis Approach** **5**
 - 2.1 From Mandatory and Optional Requirements to Core Requirements 5
 - 2.2 Template for Model Analysis 9

- 3 Model Analysis** **11**
 - 3.1 Adage 11
 - 3.2 ARBAC97 - Administrative Role Based Access Control (release 1997) 14
 - 3.3 ARBAC99 - Administrative Role Based Access Control (release 1999) 17
 - 3.4 ARBAC02 - Administrative Role Based Access Control (release 2002) 20
 - 3.5 BEA WebLogic Security Framework 23
 - 3.6 Conditional Delegation in Secure Workflows 27
 - 3.7 Context-dependent Access Control 31
 - 3.8 CoSAWoE - A Model for Context-sensitive Access Control in Workflow Environments 34
 - 3.9 Flexible Team-Based Access Control Using Contexts (C-TMAC) 39
 - 3.10 A Framework for Organisational Control Principles 43
 - 3.11 Graph-Based Formalism for RBAC 47
 - 3.12 A Model of OASIS Role-Based Access Control and its Support for Active Security 50
 - 3.13 Ponder 53
 - 3.14 Role-Based Constraints Language 2000 (RCL 2000, RSL 99) 57
 - 3.15 Role Control Center RCC 59
 - 3.16 Role Graph Model 63
 - 3.17 Task-role-based access control (T-RBAC) 67
 - 3.18 UCON_{ABC} Usage Control 71
 - 3.19 USE - UML-based Specification Environment and Object Constraint Language 74
 - 3.20 eXtensible Access Control Markup Language 78
 - 3.21 Temporal role-based access control model (TRBAC, GTRBAC, X-GTRBAC) . 82
 - 3.22 Integrated Approach to Engineer and Enforce Context Constraints in RBAC Environments (xoRBAC) 86

- 4 Recommended Models** **90**

- 5 Conclusion** **92**

1 Introduction

In workpackage 5.1 we have developed a set of requirements that will help us designing the ORKA policy management system. Our intention is to design and implement a policy administration tool for security administrators in modern enterprises with distributed systems and locations.

Within this workpackage, we first identify core requirements from the requirements set. The most important requirements will be classified, related requirements will be grouped.

After that, security models from the scientific community will be analyzed if they support the core requirements. For each model, a brief summary and an evaluation of the support of the core requirements will be presented. Each core requirement may be supported, may not be supported or may be supported with additional implementation effort within the ORKA project.

The goal of the workpackage is to find security models that can be used, in part or as a whole, as a foundation for the ORKA security model that will be designed in workpackage 5.3.

The security models to be analyzed are:

- Adage
- ARBAC97 - Administrative Role Based Access Control (release 1997)
- ARBAC99 - Administrative Role Based Access Control (release 1999)
- ARBAC02 - Administrative Role Based Access Control (release 2002)
- BEA WebLogic Security Framework
- Conditional Delegation in Secure Workflows
- Context-dependent Access Control
- CoSAWoE - A Model for Context-sensitive Access Control in Workflow Environments
- Flexible Team-Based Access Control Using Contexts (C-TMAC)
- A Framework for Organisational Control Principles
- Graph-Based Formalism for RBAC
- A Model of OASIS Role-Based Access Control and its Support for Active Security
- Ponder
- Role-Based Constraints Language 2000 (RCL 2000, RSL 99)
- Role Control Center RCC
- Role Graph Model
- Task-role-based access control (T-RBAC)
- UCON_{ABC} Usage Control
- USE - UML-based Specification Environment and Object Constraint Language
- eXtensible Access Control Markup Language

- Temporal RBAC Approaches (TRBAC, GTRBAC, X-GTRBAC)
- Integrated Approach to Engineer and Enforce Context Constraints in RBAC Environments (xoRBAC)

2 Requirement Analysis Approach

In previous work packages a huge list of functional and non-functional requirements for the ORKA architecture itself and the related security administration model within ORKA were discussed. In total we defined over 100 requirements that were classified as wish (i.e., optional) or must have (i.e., mandatory). There were lots of overlapping requirements. Therefore, in the course of this section we further discuss and derive a clustering over all identified requirements in order to classify and sort the requirements based on common affinity characteristics.

2.1 From Mandatory and Optional Requirements to Core Requirements

This section enumerates 13 Core Requirements of the ORKA administrative model. These Core Requirements are derived from the over 100 requirements of the ORKA work package 5.1. In what follows we describe each Core Requirement in detail.

CR-1 Decentralized Administration: The administration of security policies in a decentralized manner is considered a Core Requirement, due to the fact that the ORKA architecture is designed to support heterogeneous, large, and decentralized system landscapes. This requirement results in a demand for supporting several policy repositories. Policies must be distributable among these repositories. A decentralized approach allows to administer policies by administrators responsible for a specific system, department, or security context. For example consider separate policy repositories each for a different department one located in Germany and one located in France. In addition the repository located in Germany consists of enterprise wide policies that affect both departments. On the other hand it is applicable for local policies, e.g. policies that only apply to the German department, due to some German law regulations, that should not be administered by the administrator of the French department.

Related requirements:

AR5.1, AR5.8

CR-2 Check Against Enterprise Security Principles: Core Requirement 2 enfolds security requirements and regulations that are derived from a corporate security guideline, such as signing outgoing emails with a personal signature, message encryption of emails, instant messages, or short session time outs if a user does not interact with the application for a while. Such policies are an important part of the ORKA security administration model and must be either part of it or accessible in order to verify defined security policies against the enterprise security principles.

Related requirements:

AR5.7

CR-3 Separation of Administration: Besides geographic-based decentralization of policy administration in a corporate network, as stated in Core Requirement 1, the paradigm of separation of concerns must be supported for policy administration. This paradigm fosters that the ORKA administration model is able to distinguish between semantic policy domains, such as an administrator's field of activity. For instance consider a company employing three different administrators. One is responsible for the user-role assignment, another administrator is responsible for the definition of authorization policies for web services and the network infrastructure, and a third administrator is responsible for the definition of dynamic access control policies that are related to a company's business processes. Because each administrator is only concerned about his administration domain it is feasible that each take care of his own repository and is only allowed to access policies according to his duties. Therefore, the administration model must be able to define policy domains, that either correspond to organizational functional units, different semantic policy levels, such as business process policies, role policies, or general enterprise policies, such as the definition of working hours and working days.

Related requirements:

AR5.11, AR5.15, AR5.16

CR-4 Support for Compliance Rules: The administrative model must be capable of supporting compliance regulations that are dictated by governmental law institutions, e.g. Sarbanes-Oxley Act or the Health Insurance Portability and Accountability Act (HIPAA). This means the administrative model must support mechanisms to verify defined security policies against existing compliance regulations and laws and must perform compliant auditing.

Related requirements:

AR5.17

CR-5 Support for Static Constraints: The administration model must be capable of expressing mutual exclusive or conflicting entities. Therefore, it must be possible to mark entities within the administrative model, e.g. users, roles, or activities, as risky or conflicting. This allows the definition of static constraints in such a way that roles marked as conflicting can not be assigned to the same user or conflicting permissions can not be assigned to the same role at design time. This means that static constraints are already enforced at the design time of user profiles, roles, and workflows.

Related requirements:

AR5.18

CR-6 Monitoring Administrative Operations: The ORKA architecture must support an ex-

tensive auditing for each administrative activity. Such activities could be the creation, modification, and deletion of security policies. In addition, each security violation performed by an administrator, for instance violating static security policies at design-time, such as adding conflicting permissions to a single role, must be audited. This monitoring is also necessary in case the defined security policies are inconsistent. In this case, the monitoring allows to undo each administrative step until a consistent state is reached.

Related requirements:

AR5.25

CR-7 Enforce Administrative Access Rights: Based on the initial case studies made in work package 1.1. administrators should not be considered as users with unrestricted access to the policy administration interface. This could lead to fraudulent behavior by manipulating their own access rights resulting in more privileges as necessary violating the least privilege paradigm. Therefore, the ORKA administrative model must be able to define and enforce access rights for administrative activities in the same way as any non-administrative user activity. We consider administrative activities as tasks including the creation, modification, and deletion of permissions, roles, profiles, as well as the creation, modification, and deletion of constraints and their related assignment to business processes or other organizational entities.

Related requirements:

AR5.2, AR5.26

CR-8 Distributed Components and Secure Communication: As stated in Core Requirement 1 the ORKA administration model must support a decentralized and distributed administration and storing of security policies. This implies that the communication between the administrative interface, the policy repositories, the policy enforcement points, and the policy decision points is secured at design-time and at run-time. It must be possible to validate the authenticity of exchanged messages, policies queries, and policy alteration. Further, a secure communication fosters four additional security requirements besides authentication, namely integrity, confidentiality, non-repudiation, and authorization. Therefore, it must be possible to sign and encrypt policy exchange between the previously mentioned entities.

Related requirements:

AR5.44

CR-9 Support Context Information: More sophisticated access control models, such as Attribute Based Access Control or xRBAC utilize additional information beside the user's identity and the requested access method. Therefore, it must be possible to specify policy information points. These policy information points provide contextual information such as local time, system states (e.g. available CPU resources), audit trails, organizational role hierarchies, or external law regulations. It must be possible to define constraints for a policy or a set of policies in order to allow more advanced, conditional, and dynamic constraint-based access decision depending on runtime conditions.

Related requirements:

AR5.40, AR5.48

CR-10 Execute Consistency and Property Checks: The Core Requirements CR-1, CR-3, and CR-7 imply a consistency analysis at design-time. Therefore, the ORKA administration model must be capable of performing consistency checks at design-time to reveal security policy violations before they occur in a productive system environment. In addition, it must be possible to verify specified policy properties against the range of values of the utilized policy language specification.

Related requirements:

AR5.5, AR5.6, AR5.49, AR5.51

CR-11 Policy Management: This Core Requirement describes the demand for the functional support of administrative tasks, such as the definition, alteration, and deletion of policies, roles, task-role assignments, and user-role assignments. Querying and displaying recent administrative activities or effective permissions of single users and roles is also mandatory, as well as extensive auditing is part of the Policy Management Core Requirement, such as querying user-based, role-based, and policy-based activities for administrators and users.

Related requirements:

AR6.x

CR-12 Workflows: This Core Requirement demands the support of entities belonging to the application domain of workflows and business processes. To be precise it must be possible to compute business process models and to extract task and control flow related information as contextual information within the policy specification and the administrative model. In this context it should also be possible to notify about access control policies that are invalid regarding the control flow. For instance consider two tasks with a separation of duty constraint that are defined in the control flow as mutual exclusive. Therefore, there exist no execution path of the process where both tasks can be activated within the same process instance.

Related requirements:

AR7.x

CR-13 Unique Characteristics: This is a wild card Core Requirement reserved to describe some unique selling factors of an analyzed administrative model. This requirement may contain any characteristic functionality or ability that is uncommon for general security administration models.

2.2 Template for Model Analysis

Selected References:

- Authors: Title.

Description:

Short model description

Analysis of Properties:

CR-1 *Decentralized Administration*: Annotation

CR-2 *Check against Security Principles*: Annotation

CR-3 *Separation of Administration*: Annotation

CR-4 *Support for Compliance Rules*: Annotation

CR-5 *Support for Static Constraints*: Annotation

CR-6 *Monitoring Administrative Operations*: Annotation

CR-7 *Enforce Administrative Access Rights*: Annotation

CR-8 *Distributed Components / Secure Communication*: Annotation

CR-9 *Support Context Information*: Annotation

CR-10 *Execute Consistency and Property Checks*: Annotation

CR-11 *Policy Management / Use Cases*: Annotation

CR-12 *Workflows*: Annotation

CR-13 *Special Features / Remarks*: Annotations / Remarks / Items

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕⊖
CR-2	Check against Security Principles	⊕⊖
CR-3	Separation of Administration	⊕⊖
CR-4	Support for Compliance Rules	⊕⊖
CR-5	Support for Static Constraints	⊕⊖
CR-6	Monitoring Administrative Operations	⊕⊖
CR-7	Enforce Administrative Access Rights	⊕⊖
CR-8	Distributed Components / Secure Communication	⊕⊖
CR-9	Support Context Information	⊕⊖
CR-10	Execute Consistency and Property Checks	⊕⊖
CR-11	Policy Management / Use Cases	⊕⊖
CR-12	Workflows	⊕⊖
CR-13	Special Features / Remarks	⊕⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3 Model Analysis

3.1 Adage

Selected References:

- M. E. Zurko, R. Simon, and T. Sanfilippo: A User-Centered, Modular Authorization Service Built on an RBAC Foundation [ZSS99].

Description:

The Authorization toolkit for Distributed Applications and Groups (Adage) project at The Open Group Research Institute explored authorization support in distributed environments. The primary goals were to make the expression and enforcement of computer-based authorization policy easy for system administrators and application writers, to support a rich variety of mechanisms built on a role-based access control (RBAC) foundation, and to use a modular architecture that made experimentation, integration, and deployment easier. To achieve these goals, Adage was based on several overarching design principles: user-centered design, policy-neutral design, modular architecture, and roles-based design. The Adage architecture is illustrated in Figure 1. From the beginning, the user-centered design of Adage focused on its potential users, the

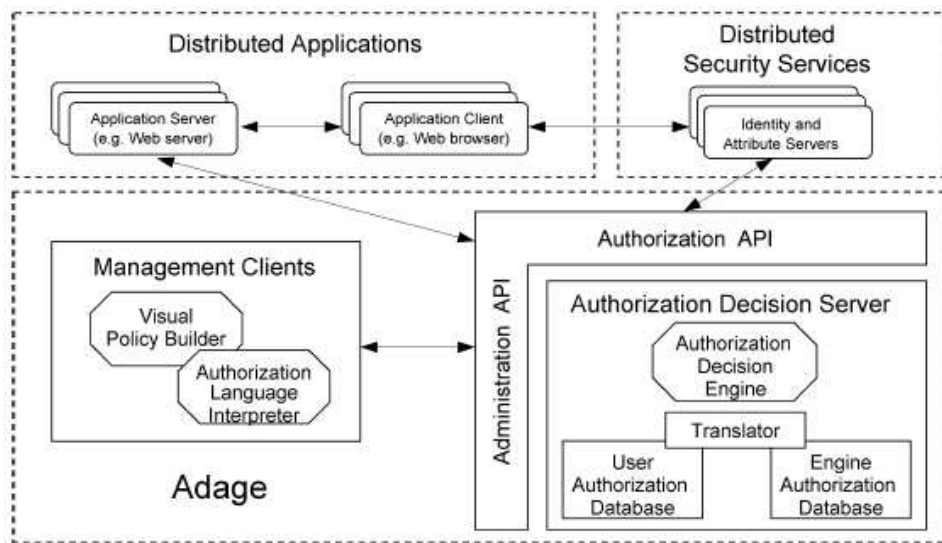


Figure 1: Adage – Conceptual Structure

creators of authorization policies for distributed enterprises. This is reflected in the design of Adage's three interfaces:

- Special emphasis is placed on the design and usability of a graphical user interface (GUI) for policy definition and management. This GUI is constructed around basic concepts that are common to all environments requiring a complex yet consistent authorization policy, such as principals, objects, groups, and rules.
- Adage also provides an alternative to the GUI for power users, in the form of a textual Authorization Language (AL) for expressing policies. The GUI and AL are equivalent in power, but the AL provides a clearer path for customization and extensions.

- Adage provides a simple application programming interface (API) for developers to query the Adage authorization server.

The approach taken to designing mechanisms was policy-neutral, involving the review of many security models, policies, and principles such as Chinese Wall Policies and Bell-LaPadula. The authors distilled the material into a set of mechanisms that could support each of the policies and principles, as well as ad hoc policies that had been observed in use. Chief among the principles supported were various forms of static and dynamic separation of duty.

Analysis of Properties:

CR-1 *Decentralized Administration*: Distributed Management Clients:

- Visual Policy Builder
- Authorization Language Interpreter

The decentralized administration components communicate via the administration API with the authorization decision server where the decision engine is placed.

CR-2 *Check against Security Principles*: Check against Security Principles is supported by implementing rules representing a site's natural security policy. Rules marked active are evaluated when an authorization request occurs at runtime.

CR-3 *Separation of Administration*: Adage does not provide explicitly separation of administration. But it is possible to define groups.

CR-4 *Support for Compliance Rules*: Support for compliance rules is supported by implementing rules and constraints representing conditions that must be met before an action can be granted.

CR-5 *Support for Static Constraints*: Support of static constraints is supported by implementing constraints representing conditions that must be met before an action can be granted.

CR-6 *Monitoring Administrative Operations*: Active rules can be displayed but are not monitored in a secure manner.

CR-7 *Enforce Administrative Access Rights*: No difference between user and administrator role hierarchies exists. User and administrator rights only can be restricted by defining various static and dynamic separation of duty constraints. According to core requirement CR-8 an authentication service should be added.

CR-8 *Distributed Components / Secure Communication*: The modular architecture of Adage allows the authorization server to work with other authentication and attribute services and secure communication infrastructures.

CR-9 *Support Context Information*: Not supported.

CR-10 *Execute Consistency and Property Checks*: Various consistency checks help ensure that the policy is not faulty but no property checks exist.

CR-11 *Policy Management / Use Cases*: Supported by the VPB as well as the AL.

CR-12 *Workflows*: Adage does not provide workflow specific features.

CR-13 *Special Features / Remarks*: User centered; modular architecture.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕⊖
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕⊖
CR-7	Enforce Administrative Access Rights	⊕⊖
CR-8	Distributed Components / Secure Communication	⊕⊖
CR-9	Support Context Information	⊖
CR-10	Execute Consistency and Property Checks	⊕⊖
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊖
CR-13	Special Features / Remarks	⊕

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.2 ARBAC97 - Administrative Role Based Access Control (release 1997)

Selected References:

- R. Sandhu, V. Bhamidipati, Q. Munawer: The ARBAC97 Model for Role-Based Administration of Roles; ACM Transactions on Information and System Security (TISSEC), Volume 2, Issue 1 (February 1999).[SBM99]

Model description:

ARBAC is an acronym for Administrative Role Based Access Control. The model release of 1997 is the first one in a series of updated specifications, the others are ARBAC99 and ARBAC02 and will be described in later chapters of this document. ARBAC97 has been developed as an RBAC extension, taking into account administrative functions that have not been specified in the original RBAC standard. The RBAC standard just deals with the relationships between users, roles and permissions, and does not specify how these objects are being created or modified. ARBAC introduces the concept of administrative roles and authority ranges to define administrative operations.

The most important RBAC goals are:

- information hiding
- least privilege
- separation of duties

Using the concept of authority ranges, the scope of the administrative operation can be defined. Each authority range contains a section within the role graph (defined as an interval), and a precondition that must be fulfilled for objects to be dealt with. In ARBAC97, five authority range types are defined:

- can-modify(-roles)
- can-assign-p(ermission)
- can-revoke-p(ermission)
- can-assign-u(ser)
- can-revoke-u(ser)

The ARBAC model defines users as autonomous persons. A role is a job function or title in an organisation, with organizational rules and responsibilities. A permission is an approval of access to one or more system objects or the grant to execute a special task. In ARBAC97 a new concept is defined consisting of three components:

- URA97: user to role assignment
- PRA97: permission to role assignment
- RRA97: role to role assignment

In the URA97 part of ARBAC97, the can-assign-u and can-revoke-u authority range define the administrative scope. Using an additional precondition, the pool of users used in an operation can be controlled.

PRA97 is similar to URA97, but handles the permission operations.

RRA97 has the can-modify authority range and defines the role operations.

Analysis of Properties:

- CR-1 *Decentralized Administration*: Decentralized Administration in the meaning of technical separation (e.g. geographical separation) is one of the most important goals of ARBAC97 and thus is supported by the model. ARBAC97 is based on the RBAC standard RBAC96. RBAC models have been defined to simplify the administration of security policies. The ARBAC97 model combines authorization management, role based access control and security administration. With support of decentralized administration an enterprise is able to administrate different locations independently.
- CR-2 *Check against Security Principles*: Check against Security Principles is supported by implementing prerequisite conditions and constraints. Prerequisite conditions can be defined for restricting administrators assigning roles and permissions. The assignment can be performed only if the preconditions are fulfilled.
- CR-3 *Separation of Administration*: Separation of Administration, in the meaning of functional separation, is supported. The definition and evaluation of preconditions is a key feature for dynamic separation of duty in ARBAC97. Using prerequisite conditions, enterprises can implement functional separation of administration, e.g. it is possible to administrate two departments with different administrators.
- CR-4 *Support for Compliance Rules*: Support for Compliance Rules is supported by implementing prerequisite conditions and constraints. ARBAC97 only provides a limited set of prerequisite conditions to define compliance rules, therefore administrators will have to deal with some restrictions in policy expressiveness.
- CR-5 *Support for Static Constraints*: Support for Static Constraints is supported by implementing prerequisite conditions and constraints.
- CR-6 *Monitoring Administrative Operations*: Monitoring Administrative Operations is not within the scope of ARBAC97. This core requirement can be seen as an addition to the ARBAC97 model.
- CR-7 *Enforce Administrative Access Rights*: Enforce Administrative Access Rights is supported by administrative role hierarchies and authority ranges. Deployment of administrative role hierarchies allows to define complex administrative access rights.
- CR-8 *Distributed Components / Secure Communication*: Distribute Components / Secure Communication is not within the scope of ARBAC97 model since it is not an implementation model. This core requirement can be seen as an addition to the ARBAC97 model.
- CR-9 *Support Context Information*: Support Context Information is not supported in a common way. Only specific constraints are defined in the model. An implementation of this core requirement seems to be possible by extending the precondition mechanisms.

CR-10 *Execute Consistency and Property Checks*: Execute Consistency and Property Checks is supported by implementing the intrinsic consistency rules of the model.

CR-11 *Policy Management / Use Cases*: Policy Management / Use Cases is not within the scope of ARBAC97. This core requirement can be seen as an addition to the ARBAC97 model.

CR-12 *Workflows*: Workflows are not within the scope of ARBAC97. This core requirement can be seen as an addition to the ARBAC97 model.

CR-13 *Special Features / Remarks*: ARBAC97 is an RBAC model that does not define any implementation details. The model can be verified by mathematical operations. Application examples for practical use are not defined. If ARBAC97 is to be used as the base of the ORKA model, workflow and enterprise aspects will have to be added in additional modules.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕⊖
CR-7	Enforce Administrative Access Rights	⊕
CR-8	Distributed Components / Secure Communication	⊕⊖
CR-9	Support Context Information	⊕⊖
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕⊖
CR-12	Workflows	⊕⊖
CR-13	Special Features / Remarks	⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.3 ARBAC99 - Administrative Role Based Access Control (release 1999)

Selected References:

- R. Sandhu, Q. Munawer: The ARBAC99 Model for Administration of Roles; Laboratory for Information Security Technology (LIST), George Mason University, Fairfax, VA; (1999).[SM99]

Model description:

ARBAC99 (Administrative Role Based Access Control - release 1999) corresponds to the ARBAC97 model but defines one extension: a distinction of mobile and immobile assignments has been specified.

As in ARBAC97, the user is granted access to all roles and thus permissions assigned to him. Likewise, the user may be granted additional roles if certain preconditions are fulfilled.

Using an immobile assignment of the user to a role, the preconditions are never fulfilled by definition, therefore the assignment of additional roles based on existing ones is restricted. If the assignment is mobile, ARBAC99 behaves like ARBAC97.

This model extension is useful e.g. for temporary visitors of a department. The assigned immobile membership avoids illegitimate access to any other domain of the enterprise.

Immobile assignment can be used for role-permission assignments as well.

ARBAC99 consists of three components (similar to ARBAC97):

- URA99: user to role assignment
- PRA99: permission to role assignment
- RRA99: role to role assignment

In URA99 the assignment of user to role is managed by the administrative roles. The extension of the ARBAC99 model with mobile and immobile has an effect. If the users membership is mobile, an administrator can assign additional roles to him. If the users membership is immobile, the administrator is not able to assign other roles, even if all preconditions are fulfilled.

Analysis of Properties:

Hint: ARBAC99 is nearly similar to ARBAC97. The only extension is the possibility to define assignments as mobile or immobile. Hence there are marginal distinctions between the evaluation of ARBAC97 and ARBAC99.

CR-1 *Decentralized Administration*: Decentralized Administration in the meaning of technical separation (e.g. geographical separation) is one of the most important goals of ARBAC99 and thus is supported by the model. ARBAC99 is based on the RBAC standard RBAC96. RBAC models have been defined to simplify the administration of security policies. The ARBAC99 model combines authorization management, role based access control and

security administration. With support of decentralized administration an enterprise is able to administrate different locations independently.

- CR-2 *Check against Security Principles*: Check against Security Principles is supported by implementing prerequisite conditions and constraints. Prerequisite conditions can be defined for restricting administrators assigning roles and permissions. The assignment can be performed only if the preconditions are fulfilled.
- CR-3 *Separation of Administration*: Separation of Administration, in the meaning of functional separation, is supported. The definition and evaluation of preconditions is a key feature for dynamic separation of duty in ARBAC99. Using prerequisite conditions, enterprises can implement functional separation of administration, e.g. it is possible to administrate two departments with different administrators.
- CR-4 *Support for Compliance Rules*: Support for Compliance Rules is supported by implementing prerequisite conditions and constraints. ARBAC99 only provides a limited set of prerequisite conditions to define compliance rules, therefore administrators will have to deal with some restrictions in policy expressiveness. The enhancement of ARBAC99 with mobile and immobile membership can be used to implement additional compliance rules, e.g. a restriction of permissions due to a special guest state.
- CR-5 *Support for Static Constraints*: Support for Static Constraints is supported by implementing prerequisite conditions and constraints.
- CR-6 *Monitoring Administrative Operations*: Monitoring Administrative Operations is not within the scope of ARBAC99. This core requirement can be seen as an addition to the ARBAC99 model.
- CR-7 *Enforce Administrative Access Rights*: Enforce Administrative Access Rights is supported by administrative role hierarchies and authority ranges. Deployment of administrative role hierarchies allows to define complex administrative access rights.
- CR-8 *Distributed Components / Secure Communication*: Distribute Components / Secure Communication is not within the scope of ARBAC99 model since it is not an implementation model. This core requirement can be seen as an addition to the ARBAC99 model.
- CR-9 *Support Context Information*: Support Context Information is not supported in a common way. Only specific constraints are defined in the model. An implementation of this core requirement seems to be possible by extending the precondition mechanisms.
- CR-10 *Execute Consistency and Property Checks*: Execute Consistency and Property Checks is supported by implementing the intrinsic consistency rules of the model.
- CR-11 *Policy Management / Use Cases*: Policy Management / Use Cases is not within the scope of ARBAC99. This core requirement can be seen as an addition to the ARBAC99 model.
- CR-12 *Workflows*: Workflows are not within the scope of ARBAC99. This core requirement can be seen as an addition to the ARBAC99 model.
- CR-13 *Special Features / Remarks*: ARBAC99 is an RBAC model that does not define any implementation details. The model can be verified by mathematical operations. Application examples for practical use are not defined. If ARBAC99 is to be used as the base of

the ORKA model, workflow and enterprise aspects will have to be added in additional modules.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕⊖
CR-7	Enforce Administrative Access Rights	⊕
CR-8	Distributed Components / Secure Communication	⊕⊖
CR-9	Support Context Information	⊕⊖
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕⊖
CR-12	Workflows	⊕⊖
CR-13	Special Features / Remarks	⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.4 ARBAC02 - Administrative Role Based Access Control (release 2002)

Selected References:

- Sejong Oh, Ravi Sandhu, Xinwen Zhang: An Effective Role Administration Model Using Organization Structure; Dankook University, George Mason University; ACM Transactions on Information and System Security, Vol.9, No. 2, May 2006.[SO06a]

Model description:

ARBAC02 (Administrative Role Based Access Control - release 2002) is based on ARBAC97. The main intention for reengineering the ARBAC97 model was to simplify the definition of users and permissions. Using ARBAC97, administrators are restricted by prerequisite conditions. Therefore, assigning a user to a role may imply multiple operations, resulting redundant role assignments.

Another weakness of ARBAC97 ensues at assignment of role to role (RRA97). Due to inheritance from the top to the bottom it is possible to assign permissions out of the defined range.

The processes of prerequisite conditions, users and permissions are refined in ARBAC02 and decoupled from roles and role hierarchies. Multistep assignments and redundancies are avoided.

ARBAC02 defines two new components, a user pool and a permission pool. Both pools are structured hierarchical. The user pool is structured as a rooted tree (from down to up) and the permission pool as a inverted rooted tree (from up to down). These pools can be modeled as organizational units.

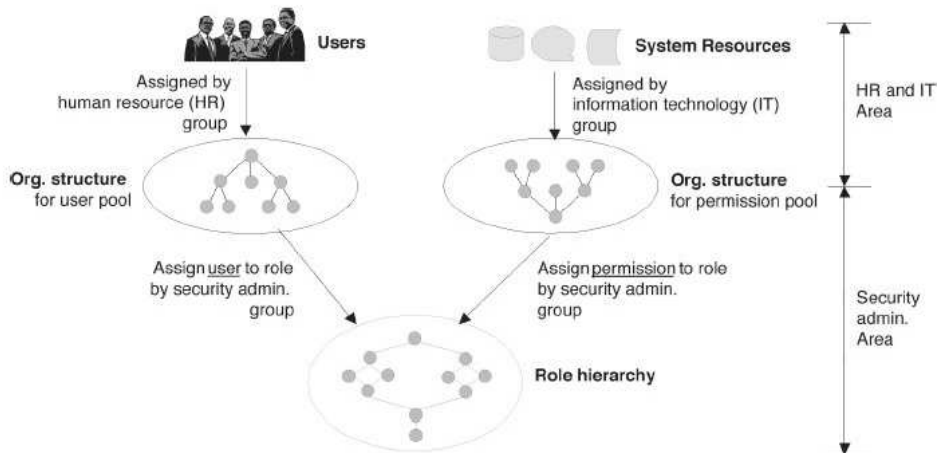


Figure 2: ARBAC02: Role administration concept

The prerequisite conditions from ARBAC97 have been replaced with the organizational structure of the user and permission pools.

Analysis of Properties:

Hint: ARBAC02 is nearly similar to ARBAC97 and ARBAC99. Only ORKA core requirement 9 is touched by the model update. CR-9 and CR-13 have changed due to the new concept of ARBAC02.

- CR-1 *Decentralized Administration*: Decentralized Administration in the meaning of technical separation (e.g. geographical separation) is one of the most important goals of ARBAC02 and thus is supported by the model. ARBAC02 is based on the RBAC standard RBAC96. RBAC models have been defined to simplify the administration of security policies. The ARBAC02 model combines authorization management, role based access control and security administration. With support of decentralized administration an enterprise is able to administrate different locations independently.
- CR-2 *Check against Security Principles*: Check against Security Principles is supported by implementing prerequisite conditions and constraints. Prerequisite conditions can be defined for restricting administrators assigning roles and permissions. The assignment can be performed only if the preconditions are fulfilled.
- CR-3 *Separation of Administration*: Separation of Administration, in the meaning of functional separation, is supported. The definition and evaluation of preconditions is a key feature for dynamic separation of duty in ARBAC02. Using prerequisite conditions, enterprises can implement functional separation of administration, e.g. it is possible to administrate two departments with different administrators.
- CR-4 *Support for Compliance Rules*: Support for Compliance Rules is supported by implementing prerequisite conditions and constraints. ARBAC02 only provides a limited set of prerequisite conditions to define compliance rules, therefore administrators will have to deal with some restrictions in policy expressiveness.
- CR-5 *Support for Static Constraints*: Support for Static Constraints is supported by implementing prerequisite conditions and constraints.
- CR-6 *Monitoring Administrative Operations*: Monitoring Administrative Operations is not within the scope of ARBAC02. This core requirement can be seen as an addition to the ARBAC02 model.
- CR-7 *Enforce Administrative Access Rights*: Enforce Administrative Access Rights is supported by administrative role hierarchies and authority ranges. Deployment of administrative role hierarchies allows to define complex administrative access rights.
- CR-8 *Distributed Components / Secure Communication*: Distribute Components / Secure Communication is not within the scope of ARBAC02 model since it is not an implementation model. This core requirement can be seen as an addition to the ARBAC02 model.
- CR-9 *Support Context Information*: Support Context Information is not supported in a common way. Only specific constraints are defined in the model. An implementation of this core requirement seems to be possible by extending the precondition mechanisms. ARBAC02 offers an additional context support with the concept of user and permission pools. It allows using organizational units as context information.
- CR-10 *Execute Consistency and Property Checks*: Execute Consistency and Property Checks is supported by implementing the intrinsic consistency rules of the model.

CR-11 *Policy Management / Use Cases*: Policy Management / Use Cases is not within the scope of ARBAC02. This core requirement can be seen as an addition to the ARBAC02 model.

CR-12 *Workflows*: Workflows are not within the scope of ARBAC02. This core requirement can be seen as an addition to the ARBAC02 model.

CR-13 *Special Features / Remarks*: ARBAC02 is an RBAC model that does not define any implementation details. Nevertheless it does define new concepts for enterprises, making the model more usable than ARBAC97. The model can be verified by mathematical operations. Application examples for practical use are not defined. If ARBAC02 is to be used as the base of the ORKA model, workflow and enterprise aspects will have to be added in additional modules.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕⊖
CR-7	Enforce Administrative Access Rights	⊕
CR-8	Distributed Components / Secure Communication	⊕⊖
CR-9	Support Context Information	⊕⊖
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕⊖
CR-12	Workflows	⊕⊖
CR-13	Special Features / Remarks	⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.5 BEA WebLogic Security Framework

Selected References:

- BEA Systems: BEAWebLogic Server: Introduction to WebLogic Security [Sys06a].
- BEA Systems: Understanding WebLogic Resource Security [Sys06b].
- BEA Systems: Using XACML Documents to Secure WebLogic Resources [Sys06c].

Description:

“BEA WebLogic Server is an enterprise-ready J2EE application server that supports the deployment of mission-critical applications in a robust, secure, highly available, and scalable environment. WebLogic Server is an ideal foundation for building applications based on Service Oriented Architectures (SOA)” [Sys06a].

The framework comprises of interfaces, classes, and exceptions to secure a service-oriented enterprise architecture. The primary function of the WebLogic Security Framework is to provide a simplified application programming interface (API) that can be used by security and application developers to define security services. A security service is initiated when an user or system process requests a protected resource on which it will attempt to perform a given operation (cf. Figure 3). The resource container that handles the type of resource being requested intercepts the request and calls the security framework and passes in the request parameters, including information such as the subject of the request and the requested resource. The security framework calls the configured role mappers and passes in the request parameters in a format that the role mappers can use. They use the request parameters to compute a list of roles to which the subject making the request is entitled and passes the list of applicable roles back to the security framework. A policy decision point (PDP) determines whether the subject is entitled to perform the requested action on the resource or not [Sys06a].

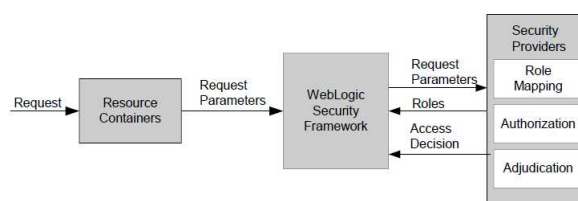


Figure 3: Resource Authorization

The BeA WebLogic framework uses security policies expressed in XACML to answer the question “Who has access to a resource?” A security policy is created to define which resource can be accessed by one or more users, groups, or security roles. These security policies are based on roles and default global groups or single users. Policies can optionally be extended by constraints, such as time constraints.

Security policies are stored in an Authorization provider’s database, for example security policies may be stored in an embedded LDAP server. WebLogic’s Authorization provider is a PDP

that actually answers the “is access allowed?” question. Specifically, an access decision is asked whether a subject has permission to perform a given operation on a resource, with specific parameters in an application. Given this information, the PDP responds with a result of *Permit*, *Deny*, or *Abstain* (i.e. there exists a conflicting authorization).

Analysis of Properties:

CR-1 *Decentralized Administration*: The BEA WebLogic Portal is a platform to provide an enterprise integration based on a service-oriented architecture. As shown in Figure 4 the framework is composed of several modules that can be stored in different locations. Because the whole WebLogic platform is realized as a portal it can be accessed by a web browser via the intranet as well as the Internet.

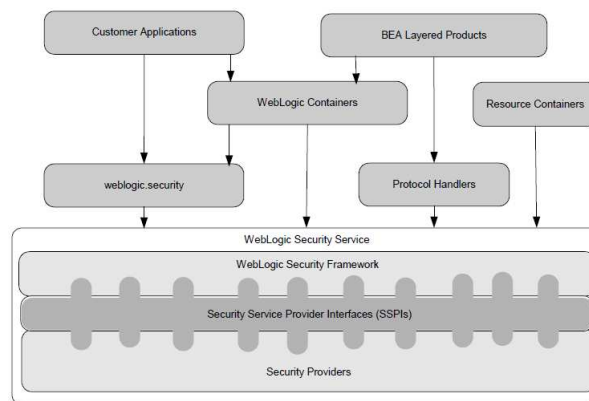


Figure 4: BEA WebLogic Architecture [Sys06a]

CR-2 *Check against Security Principles*: It is possible to define authentication and encryption requirements for protected resources. In addition, the WebLogic API can be extended to support enterprise specific security principles. Further, WebLogic defines so called (external) authorization providers that can be queried to check against corporate security principles.

CR-3 *Separation of Administration*: The WebLogic Framework supports policy and role scopes. A scope can be considered as a specific domain. Therefore, it is possible to assign administrative roles to a specific scope. This allows to restrict administrative rights to a defined scope and an administrator with a scoped role would only be able to access policies and roles within the same scope (cf. Figure 5). It should be mentioned that there is no statement about an enforcement of administrative scopes. Therefore, some modification might be necessary.

CR-4 *Support for Compliance Rules*: The WebLogic platform integrates the ILog JRules business rules engine. Therefore, compliance rules for operational activities are directly supported by WebLogic.

CR-5 *Support for Static Constraints*: The eXtensible Access Control Markup Language (XACML) is an XML language for expressing authorization policies and role assignments. The WebLogic Server XACML Authorization Provider and the WebLogic Server XACML Role Mapping Provider implement the XACML 2.0 Core Specification [Sys06c].

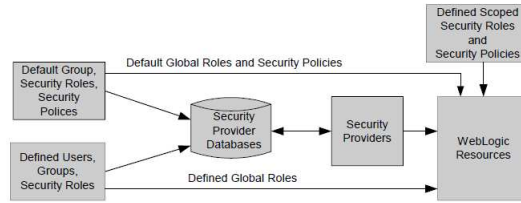


Figure 5: Global and Scoped Policies and Roles

- CR-6 *Monitoring Administrative Operations:* The administration interfaces are not considered as protected resources. Because auditing within the WebLogic framework is performed by a security module (i.e. PEP) that intercepts access requests to protected resources. There is no administrative operations monitoring available as long as the administration application itself is not considered a protected resource.
- CR-7 *Enforce Administrative Access Rights:* As stated in Core Requirement 3, it is possible to define policy and role scopes to define security and role domains. Nevertheless, policy objects and role definitions itself are not considered as protected resources. Therefore, it is not possible to define and enforce access control restrictions for them.
- CR-8 *Distributed Components / Secure Communication:* The WebLogic Framework is based on a service-oriented architecture. The communication between the different components is secured by using certificates, single-sign on, and SSL-based message security. Thus, Core Requirement 8 is directly supported.
- CR-9 *Support Context Information:* The WebLogic architecture defines dedicated context providers. Each is a class that obtains additional context and container-specific information from protected resources and provides that information to the security components making access or role mapping decisions.
- CR-10 *Execute Consistency and Property Checks:* WebLogic does not support property checks and consistency analysis. Therefore, an administrator has to be careful when defining complex security policies [Sys06b]. Notice that WebLogic uses XQuery functions that test at runtime whether an user associated with the current request context can access the specified resource, which is denoted by a element name and a resource identifier. Therefore, it is likely to enhance the administration interface to use XQuery functions already at design time to perform property checks.
- CR-11 *Policy Management / Use Cases:* The WebLogic framework provides an elaborate web-based administration interface. This interface can be used to define protected resources, user, roles, scopes, and security policies. The policy editor is shown in Figure 6. Therefore, Core Requirement 11 is directly supported.
- CR-12 *Workflows:* The WebLogic Framework does not integrate a workflow management system. It is likely that it is possible to integrate a workflow management system to cooperate with the portal and the application server, but currently it is not supported.
- CR-13 *Special Features / Remarks:* The WebLogic architecture provides a solid platform for security in service-oriented architectures. The security architecture and its integration within the WebLogic portal and application server is a good source of inspiration for the

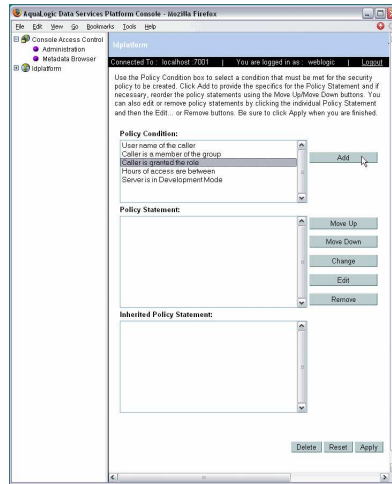


Figure 6: Administrative Policy Editing with WebLogic

realization of the ORKA architecture.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕⊖
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕⊖
CR-7	Enforce Administrative Access Rights	⊖
CR-8	Distributed Components / Secure Communication	⊕
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊕⊖
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊖
CR-13	Special Features / Remarks	⊕

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.6 Conditional Delegation in Secure Workflows

Selected References:

- V. Atluri and J. Warner: Supporting Conditional Delegation in Secure Workflow Management Systems [AW05].
- E. Bertino and E. Ferrari and V. Atluri: The Specification and Enforcement of Authorization Constraints in Workflow Management Systems [BFA99].

Description:

Atluri et al. define a framework for specifying conditions on task delegation in workflows.

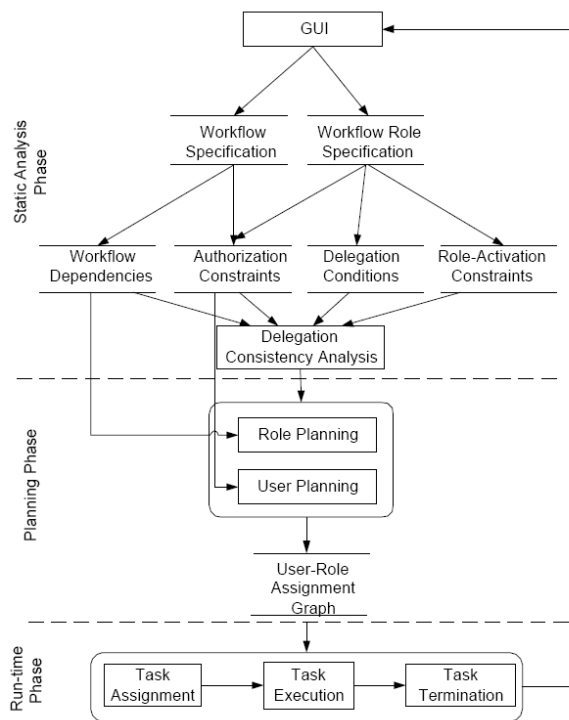


Figure 7: Assignment Phases for Workflow Execution

For this purpose, special predicates for conditional delegation are introduced as an addition to specification, execution, planning and comparison predicates. This allows to specify delegation conditions, authorization constraints, task dependency requirements and role activation constraints. All constraints are represented as clauses in a normal logic program. The logic program encoding the constraints of a given workflow is denoted as constraint base (CB). Due to the appropriate choice of definitions any CB is a stratified normal program and thus has a unique stable model (as proven by Gelfond and Lifschitz). This result helps to construct algorithms for automated consistency checking of a CB (similar to model checking). Many constraints can be checked statically, some have to be checked dynamically during the execution of the workflow. The model is based on a language for defining constraints on role and user assignment to tasks in a workflow [BFA99]. Figure 7 shows the User-Task Assignment Phases for Workflow Execution under Conditional Delegation.

Analysis of Properties:

CR-1 *Decentralized Administration*: Decentralized Administration is not supported by the Conditional Delegation in Secure Workflows model (CDiSW). There is currently no dedicated administration tool available at all.

CR-2 *Check against Security Principles*: Check against Security Principles may be provided by specifying constraints. The CDiSW model distinguishes three kinds of constraints:

- Workflow constraints
- Authorization constraints
- Role activation constraints

Authorization constraints, used to express security policy, are defined as clauses in a normal logic program. The model focuses on task delegation in workflows. Checks are provided to proof the delegation rules. Tool support for checking policies against Security Principles does not exist and would have to be added.

CR-3 *Separation of Administration*: As described in CR-1 there is currently no dedicated administration tool available at all. Separation of Administration is not supported by the CDiSW model.

CR-4 *Support for Compliance Rules*: As described in CR-2 tool support for checking policies does not exist. Neither monitoring nor recording of operations is provided by the model. Support for Compliance Rules is not supported by the CDiSW model.

CR-5 *Support for Static Constraints*: The model focuses on task delegation in a given workflow model consisting of task lists, roles authorized to execute designated tasks, and constraints and conditions. The underling model [BFA99] presents a language for defining constraints on *role assignment* and *user assignment* to tasks in a workflow. Such constraint language supports, among other functions, both static and dynamic separation of duty.

CR-6 *Monitoring Administrative Operations*: As described in CR-1 there is currently no dedicated administration tool available at all. Monitoring and recording of operations is not in the focus of the model. Monitoring Administrative Operations is not supported by the CDiSW model.

CR-7 *Enforce Administrative Access Rights*: Authorization constraints, used to express security policy, are defined as clauses in a normal logic program. Such language is not intended as an administration interface language for expressing constraints but is used internally by the system to analyze and enforce constraints. An enforcement algorithm has been proposed. An administration and a translation component would have to be added. Enforce Administrative Access Rights is not explicitly supported by the CDiSW model.

CR-8 *Distributed Components / Secure Communication*: Distributed Components / Secure Communication are not supported. The use of the CDiSW model in heterogeneous and distributed environments has been announced in [BFA99] as future work.

CR-9 *Support Context Information*: Context Information is not supported by the CDiSW model.

- CR-10 *Execute Consistency and Property Checks*: In order to provide a semantic foundation for the constraint model and to formally prove consistency, the CDiSW model presents constraints as clauses in a normal logic program. The formally defined language is enhanced by the notion of *constraint-base*, which is the logic program encoding the constraints of a given workflow for consistency issues. The constraint-base has a unique stable model as semantics. There are various algorithms for checking resp. maintaining consistency. The computational complexity of these algorithms is polynomial, but in practice they may require a high computational effort in the worst case. Inference rules do not exist. A prototype for the constraint analysis and an enforcement module exist the constraint specification language was implemented using the CORAL [RSSS94] system.
- CR-11 *Policy Management / Use Cases*: A prototype has been developed for the constraint analysis and enforcement model, which provides a graphical interface by which the user enters both workflow specification and workflow constraints and analyzes the results of each phase of the methodology. But there is no integration in an existing workflow environment and no implementation for enforcement exists so far.
- CR-12 *Workflows*: The approach provides a framework for task delegation in workflows. In particular delegation can be constrained according to various conditions, for example, based on time, workload, and task attributes. Further constraints are supported including dynamic separation of duty, role activation, and workflow dependency constraints. Many constraints can be checked statically, some have to be checked dynamically during the execution of the workflow. Workflows are supported by the CDiSW model.
- CR-13 *Special Features / Remarks*: The approach is focused on task delegation in workflows. It provides a methodology for assigning roles and users to the tasks of a given workflow according to the constraints encoded in a constraint-base. Furthermore checks are provided to proof the delegation rules. The model is interesting concerning the construction of algorithms for automated consistency checking of constraints in workflows but not in relation to a convenient administration model.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊖
CR-2	Check against Security Principles	⊕⊖
CR-3	Separation of Administration	⊖
CR-4	Support for Compliance Rules	⊖
CR-5	Support for Static Constraints	⊕⊖
CR-6	Monitoring Administrative Operations	⊖
CR-7	Enforce Administrative Access Rights	⊕⊖
CR-8	Distributed Components / Secure Communication	⊖
CR-9	Support Context Information	⊖
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕⊖
CR-12	Workflows	⊕
CR-13	Special Features / Remarks	⊕

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.7 Context-dependent Access Control

Selected References:

- Ruben Wolf and Markus Schneider: Context-dependent Access Control for Web-based Collaboration Environments with Role-based Approach [WS03].
- Ruben Wolf, Markus Schneider and Thomas Keinz: A Model for Context-dependent Access Control for Web-based Services with Role-based Approach [WSK03].

Description:

The Context-dependent Access Control model is based on the role-based access control model. This model is intended to be used within a web-enabled or Internet-enabled environment where contextual information, such as the location of a requesting subject or their used authentication mechanisms, is used to dynamically adjust security policies. The contextual dependencies and their associated security policies are expressed in terms of hierarchical roles and teams (cf. Figure 8). Depending of the actual context of a requesting subject a different role may be selected for the same subject. For instance consider the role *Internal Administrator* and *External Administrator*. The later has reduced administrative privileges. Depending on whether an administrator is accessing the system from a host in the local network or from an external host, the administrator will be dynamically assigned to one of the two roles in order to use more restricted access rights in case access is requests from an untrusted network, such as the Internet.

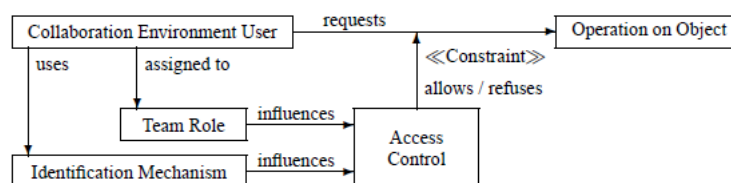


Figure 8: Context-dependent Access Control [WS03]

The Context-dependent access control model is used in the SicAri platform. It specifies a policy for all security related concerns. This policy includes statements with respect to, e.g. confidentiality, such as permissions to access some resource or the necessity to encrypt a certain communication channel. The SicAri platform fulfills most of the formulated Core Requirements of the ORKA administrative model. Please keep in mind that in what follows we only consider the access control model itself and not the whole SicAri platform.

Analysis of Properties:

CR-1 *Decentralized Administration*: The context-dependent access control model is designed to be used in web-based collaborative environments. While it is not stated directly in the related literature the usage of this framework in the UNITE and SicAri project let assume that the administration can be done by using a web-enabled application, such as a web browser. The administration interface could be accessed from any browser. Therefore, teams in the context of this model could be administrated from different locations.

- CR-2 *Check against Security Principles*: Corporate security principles can be considered as preconditions that have to be met in addition to role-based permissions. This implies that this contextual model must be able to express such preconditions. Because of this model based on the role-based access control model, it does not provide the necessary precondition entity.
- CR-3 *Separation of Administration*: The context-dependent access control model allows to define teams considered as logical groups. The context-awareness of this model can be exploited to define a fine-grained separation of administration for administrators. Consider administrator A and B, both belonging to a team A and B and both teams are located in a different location (e.g. distinct subnets). Based on the information of the utilized subnets it can be determined if an administrator belongs to team A or B and therefore is allowed to access the policies of team A or not. Thus, we consider this requirement as supported.
- CR-4 *Support for Compliance Rules*: This administrative model is based on the role-based access control mode. Compliance rules address issues, such as auditing, that are not directly related to access control. Therefore, some effort is necessary to support compliance within this model.
- CR-5 *Support for Static Constraints*: The contextual access control model is based on the classical role-based access control model. Therefore, static access control constraints are directly supported by this model.
- CR-6 *Monitoring Administrative Operations*: Based on the different kind of authentication mechanisms it could be possible to activate detailed auditing as soon as somebody logs in with strong authentication (e.g. an administrator). Using this mechanisms it could be possible to enable the desired administrative monitoring. Still, this is a conceptual model with limited tool support. Some effort to implement this feature is necessary.
- CR-7 *Enforce Administrative Access Rights*: Using the same mechanisms as described to support Core Requirement 6 it is possible to enforce administrative access rights by using the authentication mechanisms as a context provider for the actual access request. Meaning a subject with password authentication is not allowed to perform an administrative task, while a subjects authenticated by a certificate and a digital signature may be granted. Nevertheless, the described model needs some tool support to actually enforce administrative access rights.
- CR-8 *Distributed Components / Secure Communication*: The model itself does not cover implementation and communication details between a client, e.g. a web browser, the security enforcement component, and an administration interface. Because this model is intended to be used in a network or Internet environment it seems feasible to enable the support of communication security by using secure SSL connections.
- CR-9 *Support Context Information*: This contextual access control model was designed to enhance the role-based access control model by contextual information, such as a subjects used authentication method. It is likely that additional types of contextual information can be added to this model, for instance locational information or date.
- CR-10 *Execute Consistency and Property Checks*: Due to its formalization based on RBAC and its formalized description of role hierarchies is it also possible to formulate permission

queries that extract actual permissions to checks security properties and perform consistency analysis. Nevertheless, a supporting tool is missing and must be provided.

CR-11 *Policy Management / Use Cases*: Based on RBAC this access control model supports typical uses case of the role-based access control model, such as user-role assignment or permission-role assignment.

CR-12 *Workflows*: While this model is not directly incorporating process models, its RBAC-based session concept can be used to enforce strict least privilege. If we consider the duration of a task activation as a session with a dedicated role it should be possible to support this Core Requirement with some effort, but nevertheless without an awareness of the control flow this approach may error-prone.

CR-13 *Special Features / Remarks*: This is a conceptual model that lacks a real tool suite. Therefore, most of these requirements can be considered as fulfilled from a theoretical point of view. Nevertheless, a considerable effort has to be taken to fulfill all Core Requirements of the ORKA administrative model.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊖
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕⊖
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕⊖
CR-7	Enforce Administrative Access Rights	⊕⊖
CR-8	Distributed Components / Secure Communication	⊕⊖
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊖
CR-13	Special Features / Remarks	⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.8 CoSAWoE - A Model for Context-sensitive Access Control in Workflow Environments

Selected References:

- Reinhardt A Botha: CoSAWoE - A Model for Context-Sensitive Access Control in Workflow Environments [Bot01].
- Stephen Perelson et al.: SoDA: A Model for the Administration of Separation of Duty Requirements in Workflow Systems [PBE01].

Description:

The CoSAWoE model is based on the role-based access control (RBAC) model. It extends RBAC by adding the task-role-assignment relation to the RBAC model to combine workflow entities, such as tasks, task instances, and their related workflows to the RBAC model. The task-role assignment relation is used to determine which user will receive a specific task in the user’s tasklist at runtime. When a task is activated from the user’s tasklist the corresponding role assigned to the task is activated in the context of the activating user. This is indicated by the term *session control* in Figure 9. The design of CoSAWoE is strongly motivated by access control requirements, such as order of events (i.e. task assignment based on the current control flow state), strict least privilege (i.e. a task can only be activated related to the control flow), and separation of duties. Therefore, the CoSAWoE model must be deployed in a workflow environment. The complete model as described by Botha is illustrated in Figure 9.

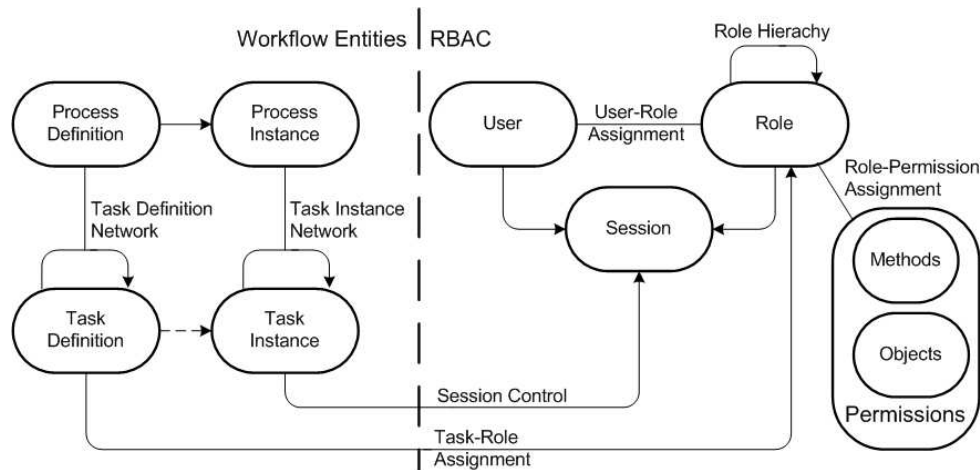


Figure 9: CoSAWoE: Integrating workflow and RBAC

SoDa and WACC Prototype:

The SoDA prototype is a tool for Separation of Duty administration as part of the CoSAWoE model [PBE01]. SoDA is a tool that focuses on supporting access control administration. Access control requirements are typically described within the general context of a business process. Therefore, SoDA is concerned with the process and task definitions and is used to define the associations between all the entities that are involved, namely user, roles, permissions, tasks,

and objects. The SoDA prototype considers an object to be a document containing various field objects. Users may perform different actions on the field objects, e.g. edit the contents or view the contents of a field. Objects may be grouped, resulting in composite objects. Permissions assigned to a composite object are inherited from objects contained by that composite object. For example, the permission to edit composite object *Employee Details* will imply the permission to edit all fields that are part of the *Employee Details* object. Object permissions are assigned to roles and roles may be related through a partial order. In order to enforce static and dynamic separation of duty, the SoDA prototype ensures that the integrity of the associations between entities is maintained. If an action cannot be performed, remedial actions are suggested. For example, if conflicting tasks are assigned to non-conflicting roles, the user is given the option of making the roles conflicting [PBE01]. WACC is a scaled-down workflow system as part of the CoSAWoE model. It provides a worklist interface whereby users can act on the tasks provided to them [Bot01]. Permissions granted depend on the order of events dictated by the underlying process description enforcing strict least privilege. To support dynamic separation of duty, users are not able to perform tasks with respect to the conflicting entities paradigm. The architecture of WACC consisting of a client, a server with a workflow system, a policy enforcement point (Access Control Module), a set of applications and access control policies stored in a database. The complete architecture is shown in Figure 10.

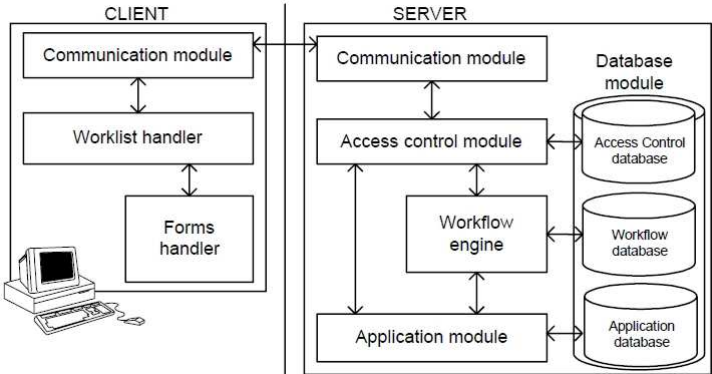


Figure 10: WACC Architecture [Bot01]

Analysis of Properties:

CR-1 *Decentralized Administration:* The WACC architecture is not designed to support remote method invocations and Web Services. The access control module is designed to access a locale access control database and protected applications are located on the same server as the other components of the WACC prototype. The WACC architecture is not designed to incorporate organizational directories such as LDAP or Active Directory. All users and roles have to be maintained within the WACC application. The distribution of user information, roles, access control policies, workflow management, and protected applications is not concerned.

CR-2 *Check against Security Principles:* The CoSAWoE model does not include generic security principles such as signatures and encryptions requirements when performing business critical tasks. It is not possible to express such generic constraints, due to the fact, that all permissions are directly related to a specific task or entity of an protected object.

CR-3 *Separation of Administration*: The SoDA prototype itself does not provide access control functionality. Therefore, each activity within SoDA can be performed by anyone. There is no authentication or authorization check integrated into the SoDA application itself. But it should be possible to use the same mechanisms as used for common protected applications in terms of protected objects, to protect SoDA as well. Therefore, SoDA activities must be expressed as tasks and related administrative forms as composite objects.

CR-4 *Support for Compliance Rules*: SoDA and WACC do not provide an auditing mechanism in order to provide necessary audit data according to compliance regulations. It is even unclear, if the scaled-down workflow management system WACC does provide a history log. No administrative activities performed in SoDA are audited by the application itself.

CR-5 *Support for Static Constraints*: SoDA is build on top of the conflicting entities paradigm that is tailored to support static conflicts between users, roles, and task. Therefore, this requirement is directly supported. The following Figure 11 depicts the definition of static conflicts between tasks in SoDA.



Figure 11: Definition of Conflicting Tasks in SoDA

CR-6 *Monitoring Administrative Operations*: As stated for CR-4. The access right management module SoDA does not provide any functionality to audit administrative activities. This functionality has to be integrated into the whole SoDA application with some considerable effort.

CR-7 *Enforce Administrative Access Rights*: Similar to CR-3, administrative activities are not access controlled. When using SoDA there is no authentication and authorization check. This means every user able to access the SoDA application may manipulate the security policies defined with SoDA. In addition, without an administrative audit trail there would not even be a trace.

CR-8 *Distributed Components / Secure Communication*: The WACC architecture does come with a secure SSL connection between the Clients and the Server. Nevertheless, without a distributed architecture (i.e. distributed policy repositories, protected applications and organizational context providers) the WACC and SoDA prototypes do not fulfill the Core Requirement CR-8 with respect to secure distributed communication between components.

CR-9 *Support Context Information*: WACC and SoDA do not consider activity based history data, organizational data, such as departments and functional units nor system information, such as CPU load or local time. Such information would be assigned to access control policies in terms of preconditions. CoSAWoE does not support any kind of precondition. Thus, context information are not part of CoSAWoE.

CR-10 *Execute Consistency and Property Checks*: The SoDA tool does runtime checks of consistency violation at design-time based on the conflicting entities paradigm. Thus, conflicting roles can not be assigned to the same user. In this case SoDA will notify the administrator. Therefore, Core Requirement 10 is directly supported.

CR-11 *Policy Management / Use Cases*: SoDA does provide all functions necessary to administrate users, roles, tasks, and permissions according to the requirements of CR-11. An example of a task-role assignment is shown in Figure 12.

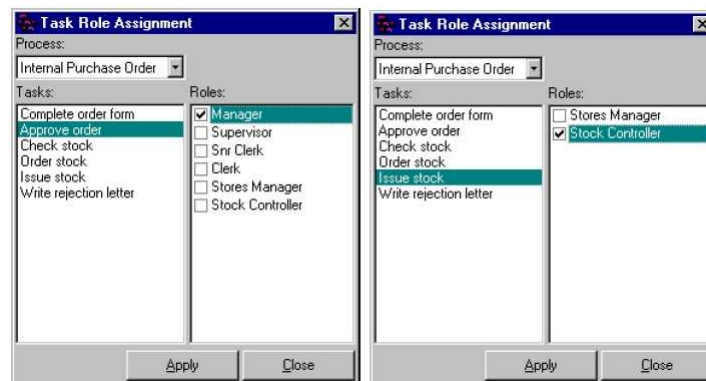


Figure 12: SoDA Role-Permission Assignment

CR-12 *Workflows*: The CoSAWoE model combines classical role-based access control with task-based access control. CoSAWoE is aware of the task entity and the order of events that make the whole workflow. Therefore, SoDA supports strict least privilege.

CR-13 *Special Features / Remarks*: CoSAWoE directly supports workflows and comes with a complete prototype collection consisting of an user client, a scaled-down workflow management system, and an administration prototype. While the CoSAWoE model and its related tools look quite promising, the centralized approach, the used technology, and programming languages are out-dated and have to be redesigned and reimplemented in order to be of some use within the ORKA project.

Evaluation of Core Requirements:

Based on the description of the previous paragraph this tables gives an overview of the overall fulfillment of the Core Requirements with respect to the CoSAWoE administrative model:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊖
CR-2	Check against Security Principles	⊖
CR-3	Separation of Administration	⊕⊖
CR-4	Support for Compliance Rules	⊖
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊖
CR-7	Enforce Administrative Access Rights	⊕⊖
CR-8	Distributed Components / Secure Communication	⊖
CR-9	Support Context Information	⊖
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊕
CR-13	Special Features / Remarks	⊕⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.9 Flexible Team-Based Access Control Using Contexts (C-TMAC)

Selected References:

- C.K. Georgiadis, I. Mavridis, G. Pangalos and R.K. Thomas: Flexible team-based access control using contexts [GMPT01].
- C. Georgiadis, I. Mavridis and G. Pangalos: Context and Role Based Hybrid Access Control for Collaborative Environments [GMP00].
- C.K. Georgiadis et al.: Implementing context and team based access control in healthcare intranets [GMNP02].
- Roshan K. Thomas: Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Control in Collaborative Environments [Tho97]

Description:

The C-TMAC (Context and Team-based Access Control) model is an active security access control model that layers dynamic access control concepts on top of RBAC (Role-based) and TMAC (Team-based) access control models.

The TMAC model was formulated by Thomas in [Tho97] to provide access control for collaborative activity best accomplished by teams of users. In TMAC, access control revolves around teams, where a team is an abstraction that encapsulates a collection of users in specific roles and collaborating with the objects of accomplishing a specific task or goal. C-TMAC does not replace roles from RBAC with teams, but extends RBAC to have both roles and teams.

Based on TMAC, C-TMAC also extends RBAC in the sense that contextual information concerning collaboration activities is associated with teams of users and user permissions dynamically filtered during runtime. These features of C-TMAC meet the specific security requirements in health care applications. Based on the technological platform of an relational data base management system and an application server, the application logic is coded with stored PL/SQL procedures that include Dynamic SQL routines for runtime value binding purposes.

The ability to integrate application level contextual information (such as date, location, patient) allows models such as C-TMAC to be flexible and express a variety of access policies that can provide tight and just-in-time permission activation. Apart from identity certificated for authentication, it uses attribute certificates for communicating critical security meta data, such as role membership and team participation of users.

There seems to be an implementation of the C-TMAC model. However, free sources for further enhancements have not been found.

Analysis of Properties:

CR-1 *Decentralized Administration*: C-TMAC extend standard RBAC with respect to teams without the need to increase the administrative overload dramatically. It is not a dedicated administration model for role-based policies. Therefore, C-TMAC does not include an administrative concept. C-TMAC does neither provide support for geographic-centralized policy descriptions nor prevents from integration of existing measures for decentralization

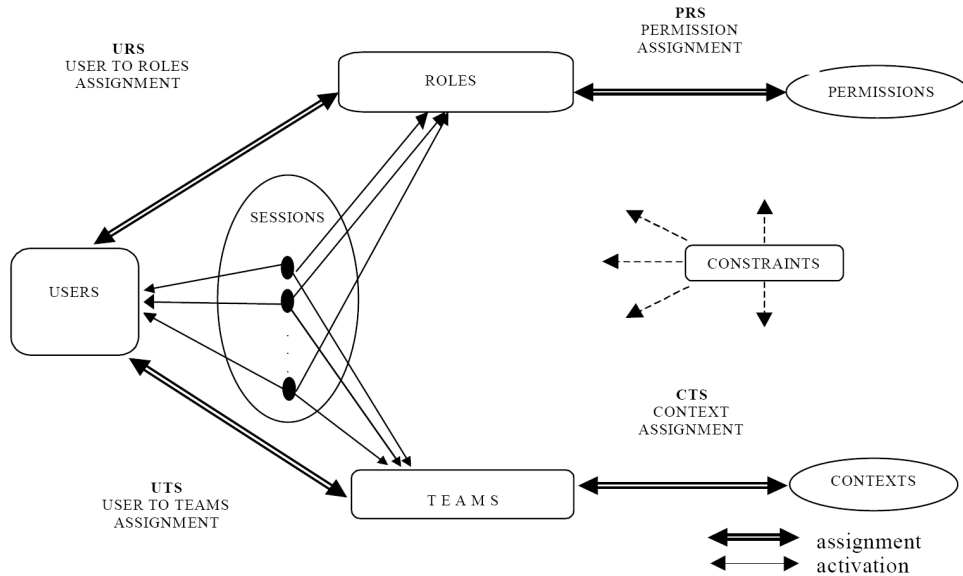


Figure 13: The C-TMAC Approach

of security policies. The underlying data model and its implementation use a relational data base management system for specification of RBAC sets and relations. All access requests are realized by SQL statements.

CR-2 *Check against Security Principles:* Even if C-TMAC is designed to support enterprises (e.g. in the health care area) it does not provide any means to check the actual policy against enterprise security principles.

CR-3 *Separation of Administration:* C-TMAC does also not provide any separation of administration features. Provision of different areas of administration and enforcement of administrative responsibilities need to be implemented out of bound. Similarly, C-TMAC does not provide any means to find inconsistencies in the policy description administrative interacting. There is no support for different administrative roles.

CR-4 *Support for Compliance Rules:* C-TMAC does not provide an auditing mechanism in order to provide necessary audit data to compliance regulations. No administrative activities performed in C-TMAC are audited by the application or model itself.

CR-5 *Support for Static Constraints:* Since C-TMAC is an extension of the standard RBAC model with respect to team support, it includes major features of standard RBAC such as static constraints. Static constraints are supported in the same way as in RBAC presumably as depicted in the architectural overview in Figure13. The C-TMAC model description does not explicitly express how constraints are handled in C-TMAC. There is an important point with respect to constraints. C-TMAC distinguishes constrains in the notion of RBAC (that may be static and dynamic) from additional dynamic contextual information (such as the actual date, location or patient). This dynamic contextual information is used (together with other information) to derive the current effective user permissions.

CR-6 *Monitoring Administrative Operations:* Since C-TMAC does not provide an administrative model and since it is based on standard RBAC which also does not provide an admin-

istrative model, monitoring of administrative operations is not supported by the model itself.

- CR-7 *Enforce Administrative Access Rights*: As mentioned before, C-TMAC does not provide an administrative model. Thus, there is no determination of different administrators. Administration of permissions may be done out of bound. The model neither requires nor supports different administrative areas (policy domains) being administered by different administrators that have been authenticated before.
- CR-8 *Distributed Components / Secure Communication*: Information about communication between the different platform components that are involved in policy processing is not given in the C-TMAC specification. The C-TMAC model does not care about, where the policy (or its parts) are stored. Instead the C-TMAC specification views policies from a logical perspective, abstracting the real location and partition of functional components into distinct and distributed system components. The model has no special support for single or multiple, local or remote policy decision and enforcement points. Of course, the C-TMAC model does not regulate how distributed components communicate in a secure way.
- CR-9 *Support Context Information*: C-TMAC supports two kinds of contextual information to be used in policy specification and policy decision. First, it provides static and dynamic constraints as proposed by the RBAC standard. Additionally there are explicit contexts assigned to teams that allow to model dynamic changing teams. Contexts are used to get the actual team membership of users and to derive the actual effective permissions of a user. In C-TMAC, just in time permission activation and revocation is realized by contexts.
- CR-10 *Execute Consistency and Property Checks*: The C-TMAC model does not focus on administrative features. It does not provide any means for execute consistency and security property checks.
- CR-11 *Policy Management / Use Cases*: The Model specifies all basic functions for policy management, since it is based on standard RBAC. Missing policy management functions may be easily realized due to the fact that all policy related information (sets and relations) are stored in a relational DBMS and may be correlated by appropriate SQL statements.
- CR-12 *Workflows*: Workflows themselves are not part of the C-TMAC specification. However, the focus of the model is on highly dynamic permissions for subjects which are part of often changing teams. Team participation is realized by modeling contexts. Here a context may be the current date, location, or patient. However, with some effort it seems reasonable to apply the notion of teams and contexts to model workflows and workflow tasks.
- CR-13 *Special Features / Remarks*: The C-TMAC is a very specialized model adding team support to role-based access control. For this reason, many of the core requirements are out of scope and cannot be supported by the C-TMAC model itself. However, the model is based on RBAC and may be enhanced with other existing RBAC extensions in order to provide a comprehensive role-based model. However, from an administrative perspective, major basic features are missing.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊖
CR-2	Check against Security Principles	⊖
CR-3	Separation of Administration	⊖
CR-4	Support for Compliance Rules	⊖
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊖
CR-7	Enforce Administrative Access Rights	⊖
CR-8	Distributed Components / Secure Communication	⊖
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊖
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊕⊖
CR-13	Special Features / Remarks	⊕⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.10 A Framework for Organisational Control Principles

Selected References:

- Andreas Schaad et al.: A Framework for Organisational Control Principles [SM02].
- Andreas Schaad: An Extended Analysis of Delegating Obligations [Sch04].

Description:

Authorizations are used to provide control mechanisms within the organization. Policies describe the underlying rules, such as who might perform some business activity. This framework for organizational control principles is a conceptual model. The modeled entities are common for any information system supporting access control. It is based on the formal specification language Alloy. Alloy is a modeling language designed to provide precise semantics for specification and modeling purposes. It is amenable to a fully automatic semantic analysis that can provide checking of consequences and consistency, and simulated execution. The following entity relationship diagram shown in Figure 14 is taken from [Sch04] and describes the entities used within the organizational control framework. The elements will be described in detail below:

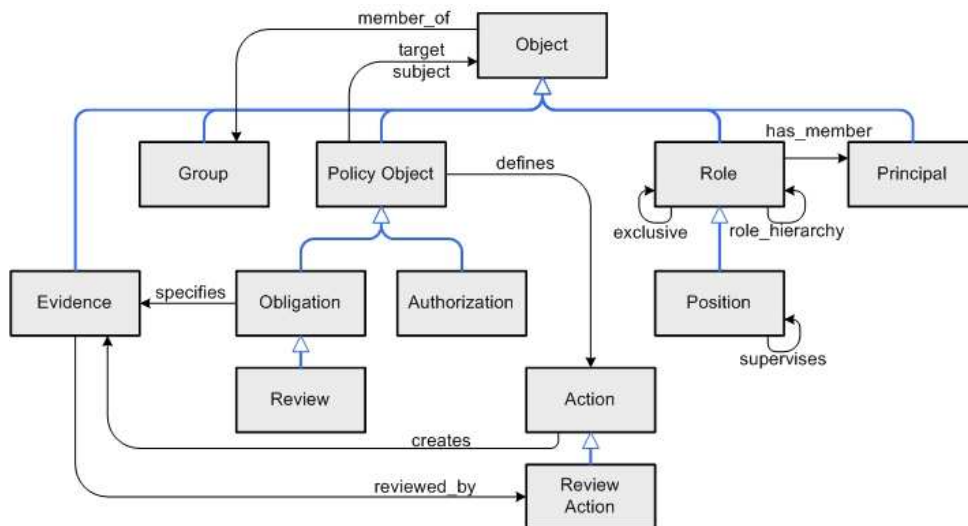


Figure 14: Organizational Control Framework

All entities are extended from a general object entity. Objects can be members of groups. Because a group is derived from an object, the group itself may be a member of another group. A principal is an object representing a human actor or an automated system component. A policy is a representation of a rule determining the behavior of principals. A policy can be differentiated into an obligation or an authorization. Policies apply to subjects, e.g. an individual principals or role, and targets. Policies define actions that have to be performed (obligation) or may be performed (authorization). Evidence is created whenever an obligated action is performed and represents some kind of investigable proof whether the action was performed or not. Review actions investigate evidences and are created when an obligation is delegated. Roles are composed out of hierarchies and mutual exclusivity. Positions are context enriched roles used to describe supervision relations through the inherited role hierarchy relation.

Analysis of Properties:

- CR-1 *Decentralized Administration*: The organizational framework itself does not state how the Alloy signatures have to be administered. The framework does not provide a dedicated model checker with specific functionalities. Decentralized administration is not a primary concern in this framework.
- CR-2 *Check against Security Principles*: In the organizational framework authorizations are always bound to actions. The underlying model does not provide an entity capable of expressing general corporate security principles, such as message encryption, message signature, or other corporate guidelines.
- CR-3 *Separation of Administration*: The proposed organizational framework does not specify a relationship between policy objects and principals and their related role and positions. Nevertheless, an extension of the organizational control framework to support the association between policy objects and principals (i.e. different administrators) seems feasible.
- CR-4 *Support for Compliance Rules*: To support compliance regulations the evidence and obligation entities could be used to specify audit regulations that can be reviewed by specifying dedicated review actions. This is not the original intention of the proposed entities, because they should be used to specify delegation and supervision paradigms. An extension of the framework is feasible.
- CR-5 *Support for Static Constraints*: The organizational framework is designed to express and support static and dynamic constraints, such as separation of duty, exclusive roles, and role hierarchies. This requirement is directly supported and can be expressed by using the proposed Alloy signatures.
- CR-6 *Monitoring Administrative Operations*: Due to the fact that the organizational control framework does not provide a policy administration interface there is currently no possibility to monitor and record any modification of the security model and the defined policy objects. Therefore, it is necessary to develop a monitored policy editor from scratch.
- CR-7 *Enforce Administrative Access Rights*: The organizational framework does not differentiate between administrative and non-administrative authorizations. Therefore, it is possible to express policy objects for administrative tasks, but without an applicable policy administration tool these constraints cannot be enforced.
- CR-8 *Distributed Components / Secure Communication*: Primarily, this framework represents conceptual work. It is not specified how components such as the model checker, the policy repository, user, and role information is realized and located. In addition, there is no information about how communication should be realized with respect to integrity, non-repudiation, or confidentiality. Due to its centralized nature this access control framework does not provide any information about message exchange and secure conversation.
- CR-9 *Support Context Information*: The organizational control framework's underlying model does support contextual constraint in the form of facts, e.g. preconditions that have to be fulfilled. The framework provides functions to evaluate contextual information, such as historic state changes that are evaluated at run-time. Therefore, it is feasible to any kind of contextual information as facts in the same way as operational or history-based preconditions that are evaluated before authorization is granted, even if it is originally not

intended by the author of the organizational control framework. The Figure 15 shows the definition of facts in Alloy.

```
//=====//
//-----Facts-----//
//=====//

//-----Constraints on groups-----//

//Initial and state-based specification of a group membership constraint
fact {all g: Group | g !in g.member_of}
fact {all s: State | all g: Group | g !in g.(s.s_member_of)}
```

Figure 15: Defining Contextual Facts

CR-10 *Execute Consistency and Property Checks*: This requirement is directly supported by the control framework. The control model and signatures defined in Alloy can be evaluated by an appropriate model checker.

CR-11 *Policy Management / Use Cases*: The proposed model directly supports the specification of roles, permissions, and conditions. While working directly within the Alloy model and performing administrative activities in Alloy most of the administrative tasks can be done by specifying Alloy constraints. The Figure 16 depicts the assignment of a principal to a role.

```
//=====//
//-----Functions-----//
//=====//

//-----Principal\Role Assignment-----//

fun assign_prin_role (disj s, s' : State, prin : Principal, r : Role) {
  (r -> prin) !in s.s_has_member &&
  s'.s_has_member = s.s_has_member + (r -> prin) &&
  prin_role_frame (s',s)
}

fun rem_prin_role (disj s, s' : State, prin : Principal, r : Role) {
  s'.s_has_member = s.s_has_member - (r -> prin) &&
  prin_role_frame(s',s)
}
```

Figure 16: Administrative Use Cases as Alloy expressions

CR-12 *Workflows*: The organizational control framework is developed with static and dynamic separation of duty in mind. Dynamic separation of duty requires a defined order of events, such as described by workflow models. While this framework does not incorporate workflow models directly, it is still possible to define sequences of tasks (i.e. authorizations). Therefore, we consider this requirement as fulfilled.

CR-13 *Special Features / Remarks*: An advantage of the organizational control framework is the complete specification of authorizations in Alloy. This allows to apply model checking, consistency analysis, and boolean authorization checks with ease. A drawback is its pure academic background with limited or insufficient tool support.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊖
CR-2	Check against Security Principles	⊖
CR-3	Separation of Administration	⊕⊖
CR-4	Support for Compliance Rules	⊕⊖
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊖
CR-7	Enforce Administrative Access Rights	⊕
CR-8	Distributed Components / Secure Communication	⊖
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊕
CR-13	Special Features / Remarks	⊕⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.11 Graph-Based Formalism for RBAC

Selected References:

- M. Koch, L. V. Mancini, F. Parisi-Presicce: A Graph Based Formalism for RBAC [KMPP02].
- H. Ehrig and G. Engels and H.-J. Kreowski and G. Rozenberg: Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages and Tools [EEKR99].

Description:

This model presents a formalization of RBAC using graph transformations that are a graphical specification technique based on a generalization of classical string grammars to nonlinear structures. The proposed formalization provides an intuitive description for the manipulation of graph structures as they occur in information systems access control and a precise specification of static and dynamic consistency conditions on graphs and graph transformations (e.g., static and dynamic SoD constraints). The formalism captures the RBAC models published in the literature, and also allows a uniform treatment of user roles and administrative roles, and a detailed analysis of the decentralization of administrative roles.

Analysis of Properties:

- CR-1 *Decentralized Administration*: The graphical specification technique of the model allows the detailed specification of various schema for decentralizing administrative roles. As an example it is possible to specify a model of decentralized administration of roles, similar to the ARBAC97 model, or to handle the revocation cascade of users membership when administrative roles are decentralized.
- CR-2 *Check against Security Principles*: By means of AGG (The Attributed Graph Grammar System) [EEKR99] it is possible to check if an RBAC configuration satisfies certain RBAC policies. Check against Security Principles is supported by specifying them as graphical constraints.
- CR-3 *Separation of Administration*: Separation of Administration is supported by specifying a graph containing the administrative roles (ar) as nodes. Each node representing an administrative role ar is connected directly by an edge with all the nodes representing the roles administered by ar. The set of roles reachable by such edges from an administrative role is the range of the administrative role.
- CR-4 *Support for Compliance Rules*: Specifying and checking of policies is provided. But neither context constraints nor history-based constraints are available. No monitoring capabilities are provided. These features would have to be added.
- CR-5 *Support for Static Constraints*: Various kinds of authorization constraints can be expressed. Graphical constraints can be used to express constraints like cardinality, mutual exclusion, prerequisite roles etc. Specification and checking of static SoD constraints is supported by defining graphical constraints.

- CR-6 *Monitoring Administrative Operations*: Monitoring Administrative Operations is not provided by the Graph-Based Formalism model.
- CR-7 *Enforce Administrative Access Rights*: There is no implementation for enforcement so far. Evaluation of constraints takes place at runtime during application of transition rules. Enforcement of Administrative Access Rights is not provided by the Graph-Based Formalism model.
- CR-8 *Distributed Components / Secure Communication*: There is no implementation for enforcement so far. Distributed Components / Secure Communication, e.g. between an administration interface and an enforcement component, is not a subject of the Graph-Based Formalism model.
- CR-9 *Support Context Information*: Various kinds of authorization constraints can be expressed. Due to the fact that AGG supports a certain kind of dynamic behavior (transformations of system states) a dynamic analysis could be possible. However, history-based authorization and time constraints are currently not supported. The effort, to add this feature by defining new types of graphic constraints, would have to be checked.
- CR-10 *Execute Consistency and Property Checks*: Consistency properties are specified by graphical constraints. By means of AGG it is possible to check if an RBAC configuration satisfies certain RBAC policies (such as SoD constraints). In addition, certain policies conflicts between authorization constraints or between authorization constraints and role hierarchies could be detected.
- CR-11 *Policy Management / Use Cases*: The basic operations of Policy Management are supported corresponding to various RBAC models like ARBAC97, the Role Graph Model, and so on. Basic operations like *add user to role*, *remove user*, *activate role*, *deactivate role*, and so on, are supported by using typed graphs. Each graph represents a state of a system. State changes are specified by graph transformation rules. A rule is formally given by a graph morphism translating the so called L(ef) graph into the R(igh) graph. Complete sets of rules for different RBAC models are given by the Graph-Based Formalism model.
- CR-12 *Workflows*: Various kinds of authorization constraints can be expressed. Due to the fact that AGG supports a certain kind of dynamic behavior (transformations of system states) a dynamic analysis could be possible. However, workflow constraints are currently not supported. The effort, to add this feature by defining new types of graphic constraints, would have to be checked.
- CR-13 *Special Features / Remarks*: The Graph-Based Formalism model presents a formalization of RBAC using graph transformations. It covers various RBAC models. General purpose graph transformation tools like AGG may be used for administration. It should be possible to translate it in a suitable form readable by ORKA's validation component in an easy way. The effort, to add the features to provide workflows and history based information, would have to be checked.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕⊖
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊖
CR-7	Enforce Administrative Access Rights	⊖
CR-8	Distributed Components / Secure Communication	⊖
CR-9	Support Context Information	⊕⊖
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊕⊖
CR-13	Special Features / Remarks	⊕

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.12 A Model of OASIS Role-Based Access Control and its Support for Active Security

Selected References:

- Walt Yao, Ken Moody, and Jean Bacon: A Model of OASIS Role-Based Access Control and its Support for Active Security [YMB01].
- D.M. Eyers, J. Bacon, and K. Moody: OASIS role-based access control for electronic health records [EBM06].

Description:

OASIS is a role-based access control architecture for achieving secure inter-operation of services in an open, distributed environment. Services define roles and implement formally specified policies for role activation and service use; users must present required credentials, in the specified context, in order to activate a role or invoke a service. Roles are activated for the duration of a session only. In addition, a role is deactivated immediately if any of the conditions of the membership rule associated with its activation becomes false (e.g., the context such as location has changed or a certain prerequisite role has been deactivated). OASIS does not use role delegation but instead defines the notion of appointment, whereby a user in some role may issue an appointment certificate to some other user. The role activation and membership conditions of services may include appointment certificates, prerequisite roles and environmental constraints.

Analysis of Properties:

- CR-1 *Decentralized Administration*: OASIS stands for Open Architecture for Secure Interworking Services. It is designed to facilitate access control in distributed systems. It embodies an open, decentralized approach; for example roles are defined by services and services may interoperate, recognizing one another's roles, according to service-level agreements.
- CR-2 *Check against Security Principles*: OASIS supports Check against Security Principles by using constraints and context dependent information. Each role is constrained by role activation rules and role membership is constraint by event triggered conditions.
- CR-3 *Separation of Administration*: Separation of Administration is supported by constraints, prerequisite roles, and the principle of appointment. One of the supported constraints is *separation of duties*. The principle of appointment allows the definition of the roles *appointer*, *appointee*, and *revoker*. Appointment implies the granting of privileges as a function of presenting credentials. For each specific type of appointment some role is identified to be the appointer. An appointee can obtain privileges by activating some role but must be known to the system. An appointment can be revoked either by its appointer only, by anyone in the appointer role, or by rule-based system revocation.
- CR-4 *Support for Compliance Rules*: Support for Compliance Rules is supported by using constraints and context dependent information. Currently the (prototype) system does not focus on providing a convenient interface for auditing. All policy decisions are recorded

into the log files at each site, but also contains a high volume of other system-related diagnostic information. Thereby compliant auditing is not explicitly performed.

- CR-5 *Support for Static Constraints*: The focus of OASIS is set on active and context dependent security. Prerequisite roles can be parameterized by context constraints as time, place, and userID and are session-based. Granting and revocation of permissions can be done by the system at runtime using an event mechanism. Role activation triggered by the principle of appointment is associated with some tasks or responsibilities a user is assign to. A user has to present his/her credentials during role activation rules are proofed. This principle would have to be modified to realize support for static constraints.
- CR-6 *Monitoring Administrative Operations*: OASIS is focused at a system level on how the policy system could be engineered and not on how policy administration would occur. As described below all policy decisions are recorded into the log files at each site, but also contains a high volume of other system-related diagnostic information. Thereby monitoring operations to enable administrative interactions is not supported.
- CR-7 *Enforce Administrative Access Rights*: Rules for activating and deactivating roles at runtime are supported by evaluating context information, i.e. time place or userID. Although this mechanisms are not explicitly developed to realize enforcement of administrative access rights it should be possible to use them by defining explicitly administrator roles.
- CR-8 *Distributed Components / Secure Communication*: As described below, OASIS is designed to facilitate access control in distributed systems. It uses HTTPS to secure connections and to communicate in a secure manner. Its prototype implementation is built over the Open Architecture for Secure Interworking Services (OASIS), an established role-based access control model developed at the Cambridge Computer Laboratory. X.509 certificates are used to authenticate users and to secure the communication between the components of the prototype. The system provides electronic health records according to the regulations of the United Kingdom National Health Service albeit in an incomplete manner.
- CR-9 *Support Context Information*: One of the main focus of the OASIS model is the specification and evaluation of context information, i.e. by parameterized roles, the principle of appointment, and role activation and deactivation triggered by (asynchronous) events.
- CR-10 *Execute Consistency and Property Checks*: OASIS uses two formal methods. The first is based on propositional logic to formalize role activation conditions. The second is based on first-order predicate calculus which allows the use of variables in expressions. This should be an adequate basis make it usable by the ORKA validation component.
- CR-11 *Policy Management / Use Cases*: OASIS does not support Policy Management / Use Cases. As described below OASIS is focused at a system level on how the policy system could be engineered and not on how policy administration would occur.
- CR-12 *Workflows*: OASIS supports workflows with its appointment model. Role activation triggered by the principle of appointment is associated with some tasks or responsibilities a user is assign to. Therefor it should be possible to define tasks and to assign them to roles. It is also possible to specify separation of duty constraints and to activate roles depending on conditions. Therefore, the principle of least privilege can be supported.
- CR-13 *Special Features / Remarks*: OASIS is focused on access control in distributed and de-

centralized systems. Dynamic aspects of role activation and deactivation characterize the model. It's qualified to be used in a web-based and client-server oriented environment. OASIS uses formal methods.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕⊖
CR-5	Support for Static Constraints	⊕⊖
CR-6	Monitoring Administrative Operations	⊖
CR-7	Enforce Administrative Access Rights	⊕
CR-8	Distributed Components / Secure Communication	⊕
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊕⊖
CR-11	Policy Management / Use Cases	⊖
CR-12	Workflows	⊕
CR-13	Special Features / Remarks	⊕

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.13 Ponder

Selected References:

- Nicodemos C. Damianou: A Policy Framework for Management of Distributed Systems; Imperial College of Science, Technology and Medicine, University of London, Department of Computing; London, February 2002.[Dam02]
- N. Damianou, N. Dulay, E. Lupu, M. Sloman: The Ponder Policy Specification Language; London, 2001. [DDLS01a]
- N. Dulay, E. Lupu, M. Sloman, N. Damianou: A Policy Deployment Model for the Ponder Language; Department of Computing, Imperial College, London; London, May 2001.[DDLS01b]
- (o. V.): The Ponder Policy Based Management Toolkit; Distributed Software Engineering Department of Computing, Imperial College, London; London, August 2002.[N.N02]

Model description:

Ponder is a language for specifying policies for management and security of distributed systems. The Ponder kit comprises a language and a toolkit for policy-based management. Main intension of Ponder is to support dynamic adaptability of management behaviour by changing policies without recording or stopping the system.

Ponder integrates thoretical practices of role-based access control (RBAC) as well as role-based management (RBM) policy specifications.

Four different basic policy types are integrated to manage Ponder:

- Authorisation policies: access of subjects (user, role) to a resource or system
- Obligation policies: event-triggered condition-action rules
- Refrain policies: define which actions a subject is not permitted to invoke
- Delegation policies: permit the subject of an authorisation policy to delegate some or all of their access rights to a new set of subjects

Furthermore Ponder possesses three composite policy types:

- roles: group basic policies which have the same subject
- relationships: policies between roles
- management structures: composition of relationships and roles

In addition Ponder offers a number of supporting abstractions to define policies:

- domains: hierarchically grouping of managed objects
- events: triggering obligation policies
- constraints: controlling the enforcement of policies at runtime

It is very important to define every single rule in the Ponder system precise and specific. The feasibility of policy implementation is fundamental for reaching enterprise goals.

Ponder is a declarative, object-oriented language for adapting enterprise rules and management policies. It is possible to use diverse access controls like firewalls, databases or Java. By keeping the language simple and by focussing on implementable policies the language can be integrated and used in management frameworks that govern the behaviour of large-scale distributed systems.

In addition to the language Ponder, a special toolkit is developed. It contains a domain browser, a compiler framework, a policy editor and a management console tool.

In the meantime the first developing process of Ponder has been terminated. Possibly some parts of the toolkit are not longer supported. A revised Ponder model (Ponder 2) has been developed. This model differs from the first model. Ponder 2 has not been taken into account.

Analysis of Properties:

CR-1 *Decentralized Administration*: Decentralized Administration is supported. The Ponder concept employs 'domains' for grouping objects with special characters. For instance grouping users due to their geographical array. Domains possess certain policies. So each domain holds references to objects within and to policies that currently apply.

CR-2 *Check against Security Principles*: Check against Security Principles is supported. For instance Ponder affords the possibility to use special security management policies to observe e.g. security violations. Ponder allows inheritances of policy types. When a type extends another, it inherits all of its elements, adds new and overrides existing elements with the same name.

CR-3 *Separation of Administration*: Separation of Administration, in the meaning of the functional separation, is supported. By means of 'domains' users can be classified in responsibility groups. Ponder integrates role-based access control (RBAC). Policies specified in terms of groups of objects are necessary for transacting separation of administration.

CR-4 *Support for Compliance Rules*: Support for Compliance Rules is supported. 'Obligation'- and 'Authorisation policies' (primitive policies) shall be used in this context. Furthermore 'composite policies' can simplify complex management policies. Grouping and structuring these policies will help to facilitate the management policies. Ponder supports access control by providing authorisation, delegation and information filtering policies called access control policies. Furthermore Ponder develops 'Delegation policies' associated with 'Authorisation policies' for specifying delegated access rights. Restrain actions of subjects are also usable by deploying 'Refrain policies'.

CR-5 *Support for Static Constraints*: Support for Static Constraints is supported. Ponder uses a subset of the Object Constraint Language (OCL) to specify constraints. Constraints are distinguished in:

- subject/target state constraints
- action/event parameters constraints

- time constraints

The deployment of 'basic'- and 'composite policies' is necessary. The instrument 'role' is very important in the field of 'composite policies'. Roles are consisting of semantic grouping of policies with a common subject. For instance a role 'service engineer' is defined with special policies.

CR-6 *Monitoring Administrative Operations*: Monitoring Administrative Operations is supported. Administrative operations are not main focus of the Ponder language. With the available items such as policies and constraints it is possible to manage administrative issues.

CR-7 *Enforce Administrative Access Rights*: Enforce Administrative Access Rights is supported. (see CR-6)

CR-8 *Distributed Components / Secure Communication*: Distribute Components / Secure Communication is supported. Distributed components are one of the main points of Ponder. Modern requirements of managing large-scale distributed systems are included by implementing certain policies. Ponder provides consistent security across distributed systems and associated legacy systems.

CR-9 *Support Context Information*: Support Context Information is supported. In this connection the item 'relationship' is very important. In Ponder relationships in the meaning of actions between different roles can be defined. The policies are grouped in rights and duties of roles towards each other. For instance, a weekly reporting is defined between the roles of a project manager and a secretary. Furthermore security policies like 'information filtering policies' for transforming interaction parameters, 'delegation policies' for granting privileges and 'refrain policies' for restraining actions are implemented.

CR-10 *Execute Consistency and Property Checks*: Execute Consistency and Property Checks are supported. Scope analysis and completeness checks inspect all policies for completeness of required elements.

CR-11 *Policy Management / Use Cases*: Policy Management / Use Cases are supported, due to the practical application of Ponder.

CR-12 *Workflows*: Workflows are supported, due to its practical application. Using 'basic' and 'composite policies' and in addition constraints and conditions, workflows can be mapped effective and simple.

CR-13 *Special Features / Remarks*: Ponder is a very practical model. By means of the definable 'basic' and 'composite policies' and their supporting abstractions 'domains', 'events' and 'constraints' all core requirements can be covered. The offering of the corresponding toolkit for simplifying the language implementation and much supporting papers to the Ponder environment are interesting too.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕
CR-7	Enforce Administrative Access Rights	⊕
CR-8	Distributed Components / Secure Communication	⊕
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊕
CR-13	Special Features / Remarks	⊕

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.14 Role-Based Constraints Language 2000 (RCL 2000, RSL 99)

Selected References:

- G.-J. Ahn, Ravi Sandhu: Role-based Authorization Constraints Specification[AS00].

Description:

The specification of security requirements and constraints in natural languages has the advantage of ease of comprehension by human beings, but may be prone to ambiguities, and the specifications do not lend themselves to the analysis of properties of the set of constraints [Ahn99]. Therefore, RCL 2000 was created with the aim to specify role-based security constraints in a unambitious fashion. RCL 2000 is a substantial generalization of RSL99 [Ahn99] and is defined in context of RBAC96. RCL 2000 is capable of defining various types of separation of duty constraints. For instance, constraints can be defined based on permissions or roles. Due to the concept of sessions, RCL2000 is also capable to express dynamic separation of duty constraints that only apply within a single session. RCL's formal approach is based on a set of expression functions that are used to determine the access rights or permission of a single user, a single role or a set of potential entities of interest. This set of functional expressions is based on formal rules. Therefore, it is possible to formally verify defined security policies and to validate if there exists inconsistencies for defined access control policies. Exclusive access rights, such as exclusive tasks or exclusive roles (e.g. used for various types of SoD constraints) are defined by the specification of so called conflict sets. RCL is based on a conceptual model and comes with formalized algorithms to query and validate access rights, even at runtime with respect to a given session. Best to our knowledge no application, i.e. policy decision point available that implements RCL2000. In addition, there is no tool available for the specification and validation of authorization constraints specified with RCL2000.

Analysis of Properties:

- CR-1 *Decentralized Administration*: The RCL2000 language does not state how the RCL2000-based constraints are stored or administered. Therefore, we consider decentralized administration as not supported.
- CR-2 *Check against Security Principles*: Because RCL is based on RBAC no contextual information used as preconditions can be expressed within RCL2000.
- CR-3 *Separation of Administration*: RCL provides no tool support for the specification of authorization policies. Further, the related entity relationship diagram does not contain specification of teams or departments. A workaround would be to define different conflicting administrative roles and associate exclusive permissions with each administrative role. Nevertheless, this comes with some effort.
- CR-4 *Support for Compliance Rules*: Because RCL is based on RBAC no contextual information, such as compliance rules, used as preconditions can be expressed within RCL2000.
- CR-5 *Support for Static Constraints*: Static separation of duty is supported by RCL2000 and can be expressed by the constraint language.

- CR-6 *Monitoring Administrative Operations*: RCL is a theoretical model only. No monitoring capabilities are provided.
- CR-7 *Enforce Administrative Access Rights*: RCL is a theoretical model only. There exists no tool support for the specification of authorization constraints with RCL2000. Thus, administrative access rights cannot be enforced.
- CR-8 *Distributed Components / Secure Communication*: RCL is a theoretical model only. There exists no proposal what environments and system landscapes this model might be suitable.
- CR-9 *Support Context Information*: RCL2000 and its defined functional expressions do not address the description of context enriched constraints.
- CR-10 *Execute Consistency and Property Checks*: RCL2000 is based on a formal model and provides algorithms to validate and verify defined authorization policies.
- CR-11 *Policy Management / Use Cases*: RCL2000 is based on RBAC96. Thus, it is possible to define users, roles, user-role assignments, and role-permission assignments. Nevertheless, this must be done by hand, because no tool support is available.
- CR-12 *Workflows*: The session concept of RCL2000 could be used to represent and define policies in the context of a process instance. Nevertheless, there exists no support for workflow based entities, such as tasks, or control flow related information.
- CR-13 *Special Features / Remarks*: RCL2000 is a pure role-based access control model. Its strength lies in the formal approach and the resulting validation and verification capabilities. There exists no tool support and unique characteristics that are not covered by any other model. Therefore, we consider this model as non-suitable within the ORKA project.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊖
CR-2	Check against Security Principles	⊖
CR-3	Separation of Administration	⊕⊖
CR-4	Support for Compliance Rules	⊖
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊖
CR-7	Enforce Administrative Access Rights	⊕⊖
CR-8	Distributed Components / Secure Communication	⊖
CR-9	Support Context Information	⊖
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕⊖
CR-12	Workflows	⊖
CR-13	Special Features / Remarks	⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.15 Role Control Center RCC

Selected References:

- D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli: Role-Based Access Control [FKC03].
- D. F. Ferraiolo, G-J. Ahn, R. Chandramouli, and S. I. Gavrila: The Role Control Center: Features and Case Studies [FACG03].

Description:

The Role Control Center (RCC) is a tool that enables centralized management of authorizations for resources distributed throughout an enterprise. For this purpose RCC provides an interface through its Client to create and maintain an enterprise-level Role-based Access control Model (ERBAC) on the RCC server and map the authorization data contained in the ERBAC model to the various target systems through the resident RCC Agent module. The ERBAC model in RCC provides support for defining arbitrary role structures (generalized role hierarchies) and flexible separation of duty constraints. RCC supports an ERBAC model that includes features such as general role hierarchy, static separation of duty constraints, and an advanced permission review facility (as defined in NIST's proposed RBAC standard). The present version of RCC contains agents for Windows NT and Apache Web Server platforms. Work is underway to develop agents for more target platforms and to enhance the policy support capabilities of the ERBAC model.

Analysis of Properties:

CR-1 *Decentralized Administration*: Decentralized Administration is supported by RCC using a client / server architecture (see also Figure 17).

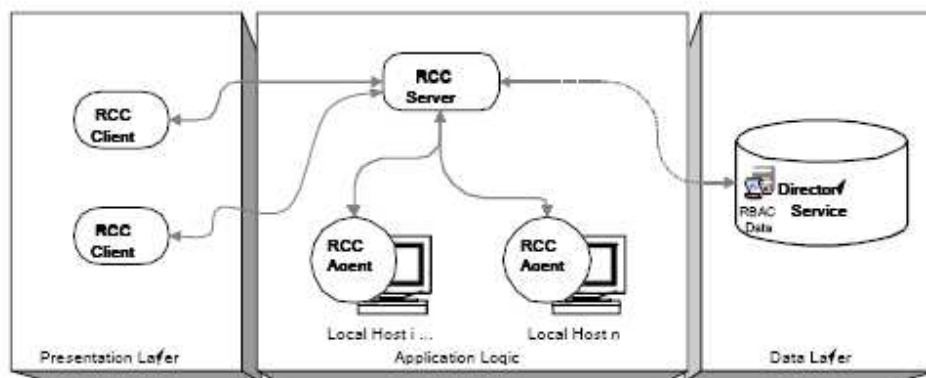


Figure 17: Role Control Center (RCC): Conceptual Structure

The RCC clients provide a graphical user interface to create and maintain the so called *Enterprise RBAC model* elements and relations (usually called *role graph*). The clients also provide the code responsible for displaying the role graph, capturing user actions and transforming them into calls to the APIs provided by the RCC server. The RCC server analyzes the commands received from the clients, retrieves the necessary data from a

central Directory Service and checks the commands' consistency with the data. The server updates the RBAC role graph informations and sends them back to the directory. The RCC server is also responsible for mapping selected subgraphs called *views* to user accounts and groups on heterogeneous distributed hosts. RCC uses agent software running on each host to create/delete groups and user accounts, populate the groups with user accounts, and set up ACLs.

- CR-2 *Check against Security Principles*: Check against Security Principles is not supported explicitly. The RCC server analyzes the commands received from the RCC clients providing the graphical administration interface and checks the commands' consistency with the data stored in a central directory. Considered an initial security policy, representing the enterprise security principles, that is stored in the central directory it should be possible to detect violations of this principles.
- CR-3 *Separation of Administration*: The RCC tool is particularly developed to support authorization administration tasks at the enterprise and system level. The permissions associated with administrative roles pertain to administrative operations like *Assign users to roles*, *Assign a role to role* (to build a hierarchy), or *Assign permissions to a role*. These operations are differentiated from resource-level operations like *Read*, *Write* etc. All the operations are carried out using menus in RCC's administrative interface. It is possible to define subgraphs for particular hosts whereby it should be possible to define disjointed administrator domains.
- CR-4 *Support for Compliance Rules*: It should be possible to define compliance rules in an initial security policy and to check administrative operations against them (see CR-2). But there is no statement about monitoring all administrative operations and to store them in a secure manner. This means that the RCC administrative model does not support mechanisms to perform compliant auditing (see CR-6). Some effort is necessary to realize it in the ORKA project.
- CR-5 *Support for Static Constraints*: Support for Static Constraints is supported by RCC providing static separation of duty constraints.
- CR-6 *Monitoring Administrative Operations*: Monitoring Administrative Operations is supported by RCC providing an advanced permission review facility. But there is no statement about monitoring all administrative operations and to store them in a secure manner. Some effort is necessary to realize it in the ORKA project.
- CR-7 *Enforce Administrative Access Rights*: There is no statement about using passwords or certificates to verify the authorization of administrators after assigning a user to an administrator role. Enforce Administrative Access Rights is not supported by the RCC model.
- CR-8 *Distributed Components / Secure Communication*: Distributed Components / Secure Communication is supported. An RCC client communicates with an RCC server and its agents using the Secure Sockets Layer (SSL) protocol. Communication between the RCC server and the Directory Service should also be secured using an analog protocol.
- CR-9 *Support Context Information*: RCC supports the creation and modification of general role hierarchies and provides the ability to define subgraphs for particular hosts (location). But it does not support context-sensitive restrictions that depend on information available only

at runtime. This means that the RCC administrative model does not support evaluation of context information.

CR-10 *Execute Consistency and Property Checks*: As described in CR-2 the RCC server analyzes the commands received from the RCC clients providing the graphical administration interface and checks the command's consistency with the data stored in a central directory. There is no statement about carrying out property checks. Considering the graphic-centric form of the RCC model it should be possible to transform the role graphs into a logical form that can be used by ORKA's validation component.

CR-11 *Policy Management / Use Cases*: The RCC tool supports Policy Management / Use Cases by providing the creation and maintenance of the role graph elements (users, roles, permissions, and constraints) and relations (user-to-role, role-to-role, and role-to-permission). The graphic interface provides predefined roles, regular roles like *payroll clerk* as well as administrative roles like *financial admin*. It also provides the mapping of the defined rules and relations to real resources and operations on system level (see also CR-1).

CR-12 *Workflows*: To support workflows it should be possible to pose restrictions based on the state of a workflow and to define a chronological order of workflow tasks. In this sense RCC does not support workflows.

CR-13 *Special Features / Remarks*: The RCC model is a relatively simple graphic-centric model. The RCC server was implemented 2003 as a Windows NT/2000 application in two versions one written in Visual Basic and the other in Java. It contains agents for Windows NT and Apache Web Server platforms. The RCC client performs a function similar to a browser in a web application but is more sophisticated. All operations performed by an RCC client are carried out by using menus. The graphical administrative interface supports an advanced permission review facility. Using the core functionality of the model it should be possible to add the missed functions in the ORKA project.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕⊖
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕⊖
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕⊖
CR-7	Enforce Administrative Access Rights	⊖
CR-8	Distributed Components / Secure Communication	⊕
CR-9	Support Context Information	⊖
CR-10	Execute Consistency and Property Checks	⊕⊖
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊖
CR-13	Special Features / Remarks	⊕

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.16 Role Graph Model

Selected References:

- Mei Ge and Sylvia Osborn: A Design for Parameterized Roles. In Data and Applications Security XVIII – Status and Prospects, Kluwer, 2004 [GO04].
- Lingling Hua and Sylvia Osborn: Modeling UNIX Access Control with Role Graph. Proceedings of International Conference on Computers and Information, June, 1998 [HO98].
- C.M. Ionita and S.L. Osborn: Privilege Administration for the Role Graph Model. Proc. IFIP WG11.3 Working Conference on Database Security, July 2002 [IO02].
- M. Nyanchama and S.L. Osborn: Access Rights Administration in Role-Based Security Systems. Database Security VIII: Status and Prospects, Biskup, Morgenstern and Landwehr, 1994 [NO94].
- S. Osborn and Y. Guo: Modeling Users in Role-Based Access Control. Fifth ACM Workshop on Role-Based Access Control, Berlin, July 2000 [OG00].
- Matunda Nyanchama and Sylvia Osborn: The Role Graph Model. First ACM Workshop on Role-Based Access Control, Gaithersburg Maryland, Nov. 1995 [NO95].
- S.L. Osborn: Information Flow Analysis of an RBAC System. Proc. ACM SACMAT, June 2002 [Os02a].
- S. Osborn, Y. Han, and J. Liu: A methodology for managing roles in legacy systems. In Proceedings of the 8th ACM Symposium on Access control models and technologies (SACMAT), pages 33–40. ACM, Jun 2003 [OHL03].
- Wang, J. and S.L. Osborn A Role-Based Approach to Access Control for XML Databases. In Proc. ACM SACMAT, June, 2004 [WO04].
- He Wang, Sylvia L. Osborn: Delegation in the role graph model, SACMAT 2006 [WO06].
- Yunyu Song, Sylvia L. Osborn: Conflict of Interest in the Administrative Role Graph Model. Secure Data Management 2006 [SO06b].
- Matunda Nyanchama and Sylvia Osborn: The role graph model and conflict of interest. ACM Transaction on Information and System Security, 2(1):3–33 1999 [NO99].
- Sylvia L. Osborn: Integrating role graphs: a tool for security integration. Data Knowl. Eng. 43(3): 317-333 (2002) [Os02b].
- Wang, H. and S. L. Osborn. An Administrative Model for Role Graphs. Proc. IFIP WG11.3 Working Conference on Database Security, Estes Park, Colorado. 2003 [WO03]

Description:

The Role Graph Model is a general access control model which allows easier management of the assignment of permissions to users when there are very large numbers of both. It groups permissions into roles, so that by assigning a user to a role, one can assign an arbitrary number of permissions at once. It also encompasses a Group Graph Model which allows users to be put

into groups, which are simply sets of users. The Role Graph Model has similar capabilities to RBAC models of Sandhu and others.

The operations available in a role graph are comparable with the standard RBAC operations. Role graphs have some special properties. For example, they have a single max role and a single min role, and there are no cycles within the role graph, i.e. the role hierarchy is represented by a single connected acyclic graph.

The Role Graph Model was proposed in 1995. Since then various extensions have been published, e.g. with respect to separation of administration, delegation, and parameterized roles.

The basic Role Graph Model was initially designed to manage permissions of a relational database system. However, it has been extended to be integrated into legacy systems in general [OHL03], UNIX file systems [HO98] or XML databases [WO04].

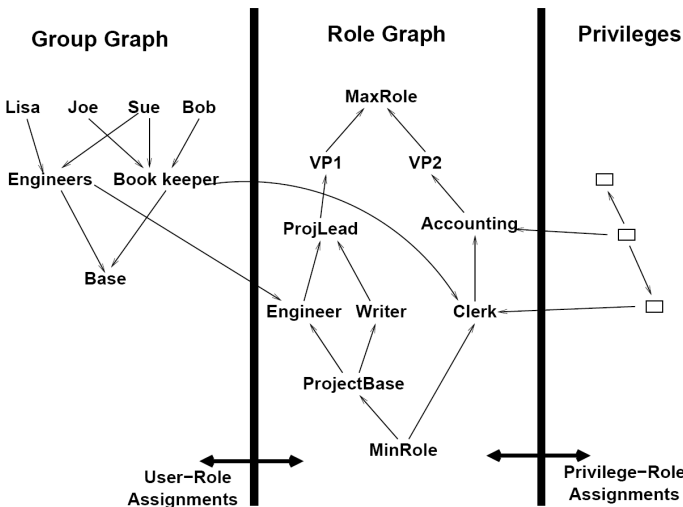


Figure 18: The Role Graph Model including a Group Graph

Analysis of Properties:

CR-1 *Decentralized Administration:* The Role Graph Model introduces a role hierarchy with special properties, e.g. with respect to a single min role and a single max role connection for all roles of a role-based security policy. For this reason, the role hierarchy is always a single connected acyclic graph. The basic Role Graph Model does not provide geographically separated sub-policies.

However, since the role graph represents the logical view on the security policy, decentralized administration of geographically distributed policy-domains may be realized, particularly since the Role Graph Model supports different administration domains (see CR-3). Additionally, in [Osb02b] Osborn shows how to merge two role graphs with overlapping permissions.

CR-2 *Check against Security Principles:* The Role Graph Model does not provides measures for checking the policy agains given security principles. However, it is possible to illustrate information flows based on a given policy description in Role Model presentation [Osb02a].

CR-3 *Separation of Administration*: The basic Role Graph Model has been extended by a comprehensive administration model with support for multiple administrative domains that are managed by different administrators with varying rights. See [SO06b, WO03, WO06].

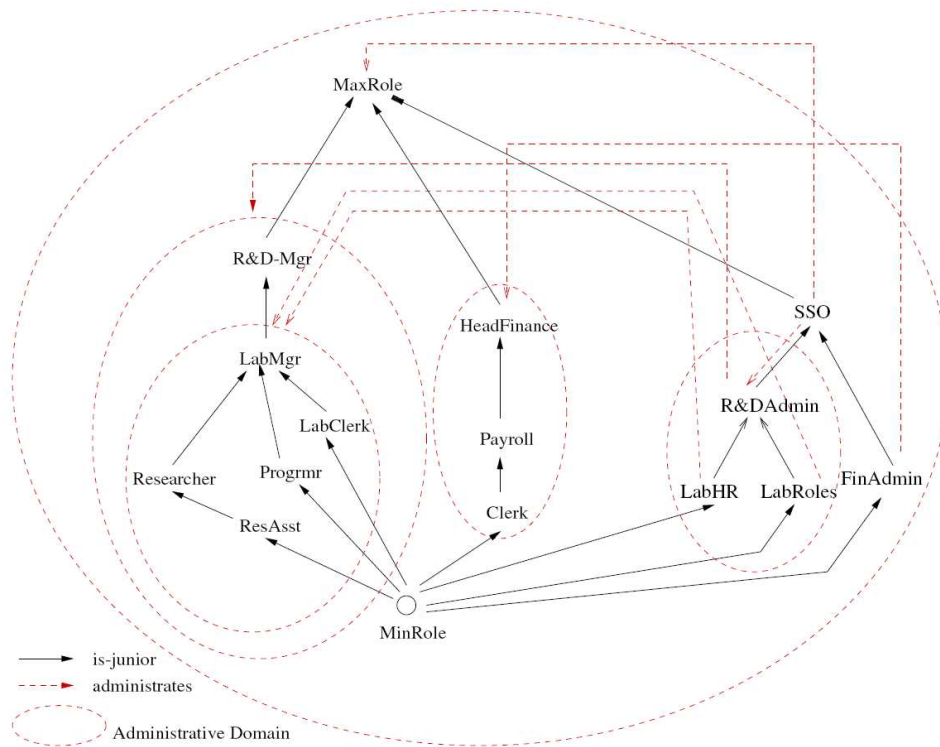


Figure 19: Administrative Domains in Role Graph Model

There is a comparison of Role Graph Model's administrative model with other approaches in [WO03].

CR-4 *Support for Compliance Rules*: The Role Graph Model does not provide any means to monitor administrative operations, which would be an information base to check compliance rules.

CR-5 *Support for Static Constraints*: The Role Graph Model supports static constraints for specification of separation of duty properties. General static constraints are not supported.

CR-6 *Monitoring Administrative Operations*: The Role Graph Model itself does not provide any means to monitor administrative operations. These features need to be realized out-of-bounds.

CR-7 *Enforce Administrative Access Rights*: The Role Graph Model supports privileges of administrators for different policy domains. Privileges are modeled within the Role Graph (see dashed lines in Figure 19). Administrators need to authenticate before managing security policies.

CR-8 *Distributed Components / Secure Communication*: The Role Graph Model itself does not specify any features for secure communication of distributed components. The model more or less specifies a logical view on local non-distributed security policies. Secure communication between distributed components need to be realized out-of-bound.

- CR-9 *Support Context Information*: The Role Graph Model does not support context information in general. It provides means to have parameterized roles. For example, if a policy needs to specify that each subject is allowed to access its own data record, the corresponding role could use the user's identity as parameter within the role specification. Additionally, there is the possibility to specify conflicting roles that could be used to model static separation of duty constraints. Dynamic constraints cannot be modeled by the Role Graph Model itself, but may be added by modifying the definition of a permission tuple into a permission triple containing the additional constraint.
- CR-10 *Execute Consistency and Property Checks*: Consistency and property checks are not within the scope of the Role Graph Model. However, it is possible to derive the information flow channels from a specified role graph policy.
- CR-11 *Policy Management / Use Cases*: The Role Graph Model supports a variety of policy management functions. Implementation details and algorithms can be found in [NO94, Osb02b]. There is a graphical user interface for policy managing [OHL03]. In [NO99], solutions for modeling conflict of interest scenarios are given.
- CR-12 *Workflows*: The Role Graph Model itself does not provide any workflow features, even if they may be modeled by using constraints or parameterized roles (see [GO04]).
- CR-13 *Special Features / Remarks*: The Role Graph Model also provides a delegation concept [WO06], parameterized roles (as an alternative to private roles for modeling a bunch of users with similar permissions) [GO04]

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕⊖
CR-2	Check against Security Principles	⊖
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊖
CR-5	Support for Static Constraints	⊕⊖
CR-6	Monitoring Administrative Operations	⊖
CR-7	Enforce Administrative Access Rights	⊕
CR-8	Distributed Components / Secure Communication	⊖
CR-9	Support Context Information	⊕⊖
CR-10	Execute Consistency and Property Checks	⊕⊖
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊖
CR-13	Special Features / Remarks	⊕

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.17 Task-role-based access control (T-RBAC)

Selected References:

- Li Zhang, Lili Luo, Liyong Zhang, Tiesuo Geng, Zongge Yue: Task-Role-Based Access Control in Application on MIS; School of Electronics and Information Engineering, University Assets Administration Office, School of Hospital; Dalian University of Technology; (Asia Pacific Conference on Service Computing), 2006.[ZLZ⁺06]
- Sejong Oh, Seog Park: Task - role-based access control model. Department of Computer Science, Sogang University, Seoul, South Korea; May 2002.[SO02]

Model description:

T-RBAC is an acronym for task-role-based access control. It is based on traditional access control models like discretionary access control (DAC), mandatory access control (MAC) and role-based access control (RBAC). The model of activity-based access control (ARBAC) influences T-RBAC too.

T-RBAC combines characteristics of enterprise environment with requirements of access control. The task-role-based access control model is mainly developed for industrial companies which have task oriented operational processes and workflows.

Main item is the integration of tasks in the core concept of the access control model. A user has a relationship with permission through role and task.



Figure 20: T-RBAC basic approach

Roles and tasks are defined as follows: The concept 'role' focuses on actor, the concept 'task' focuses on activity. In T-RBAC permissions such as access rights are assigned to tasks. Task is a unit of business work. Business role or job function can be defined as a set of tasks.

The two major factors, organisation structure and business process, which specify an enterprise environment, are integrated in the T-RBAC model. Due to the organisation structure tasks are classified in inheritable and non-inheritable tasks. Tasks belonging to the business process are classified in active tasks. Other tasks not belonging to the business process are called passive tasks.

By characteristics of organisation structure and business process four classes of tasks are defined:

- Class P (Private): access rights are not inherited and they do not belong to the business process (passive). Tasks can be e.g. analysis, planning, decision making.

- Class S (Supervision): access rights are inherited but they do not belong to a business process (passive). In this case tasks can be e.g. review, audit, monitoring, approval, delegation.
- Class W (Workflow): access rights are not inherited but they belong to a business process (active). For instance tasks in workflows like 'check product stock'. The tasks in class W have several attributes such as activation condition, duration, activation cardinality.
- Class A (Approval for activity): access rights are inherited and they belong to the business process such as workflows. E.g. tasks related with approval in the workflow.

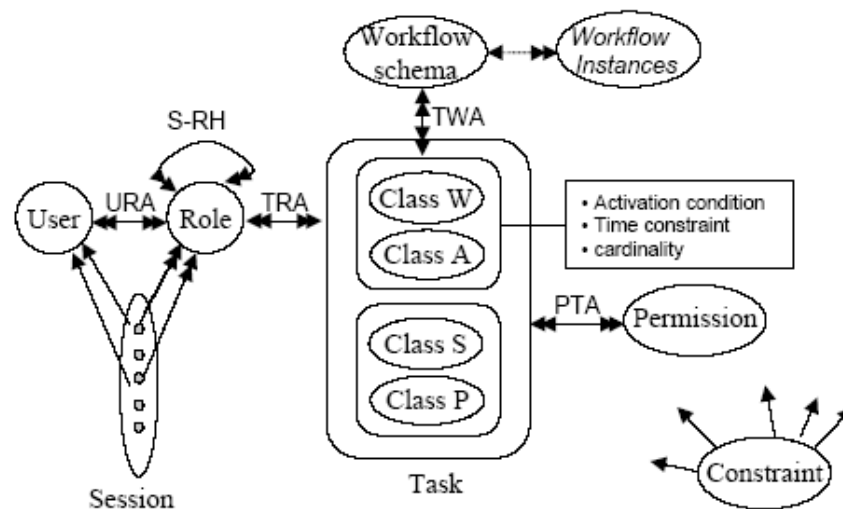


Figure 21: T-RBAC model

Analysis of Properties:

CR-1 *Decentralized Administration*: Decentralized Administration is supported. RBAC (role based access control) i.a. is covered by T-RBAC. Thus decentralized administration is a main issue of T-RBAC.

CR-2 *Check against Security Principles*: Check against Security Principles is supported by implementing constraints and conditions.

CR-3 *Separation of Administration*: Separation of Administration, in the meaning of functional separation, is supported. Delegation can be made on the base of task-based unit (instead of role-based unit). Task unit has more small scope of access rights than role unit. So T-RBAC can support more elaborate separation of duty policies.

CR-4 *Support for Compliance Rules*: Support for Compliance Rules is supported by constraints and the classified tasks. Active tasks of classes W and A can be defined with special conditions, time constraints and cardinality.

CR-5 *Support for Static Constraints*: Support for Static Constraints is supported by implementing prerequisite conditions and constraints.

- CR-6 *Monitoring Administrative Operations*: Monitoring Administrative Operations is not within the scope of the T-RBAC model. This core requirement can be seen as an addition to the T-RBAC model.
- CR-7 *Enforce Administrative Access Rights*: Enforce Administrative Access Rights is not within the scope of the T-RBAC model. This core requirement can be seen as an addition to the T-RBAC model.
- CR-8 *Distributed Components / Secure Communication*: Distribute Components / Secure Communication are not within the scope of the T-RBAC model since it is not implementation model. This core requirement can be seen as an addition to the T-RBAC model.
- CR-9 *Support Context Information*: Support Context Information is supported by constraints and the classified tasks.
- CR-10 *Execute Consistency and Property Checks*: Execute Consistency and Property Checks are supported by implementing the defined consistency rules and properties of the classified tasks of the model.
- CR-11 *Policy Management / Use Cases*: Policy Management / Use Cases are not within the scope of the T-RBAC model. This core requirement can be seen as an addition to the T-RBAC model.
- CR-12 *Workflows*: Workflows are featured in the T-RBAC model. Due to the classified tasks and the deployment of classes 'W' (workflow) and 'A' (approval for activity), workflows can be mapped by this model. In class 'W' tasks can be added with attributes like activation condition, duration and activation cardinality. Activation condition is a special precondition that a specific task has to be activated. Attribute duration restricts the activated task only for a special period. Activation cardinality is the maximum number of activations of a specific task at the same time.
- CR-13 *Special Features / Remarks*: It is another theoretical model with some approaches to modern enterprises. Tips for implementation and hints for little supporting tools are approached indeed. But the adoption of all practical processes of an enterprise is still missing.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕⊖
CR-7	Enforce Administrative Access Rights	⊕⊖
CR-8	Distributed Components / Secure Communication	⊕⊖
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕⊖
CR-12	Workflows	⊕
CR-13	Special Features / Remarks	⊕⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.18 UCON_{ABC} Usage Control

Selected References:

- Jaehong Park, George Mason University and Ravi Sandhu, NSD Security and George Mason University: The UCON_{ABC} Usage Control Model; Fairfax VA, 2004. (ACM Transactions on Information and System Security, Vol. 7, No. 1, February 2004, Pages 128-174).[PS04]

Model description:

UCON_{ABC} is an acronym for Usage CONtrol, the annex stands for Authorizations, oBligations and Conditions.

UCON is an extension of traditional access control for covering modern requirements like authorizations, obligations, conditions, continuity and mutability. It combines traditional access controls like mandatory, discretionary and role based access control as well as modern trust management and digital rights management (DRM). The digital environment requires new and enhanced security models. Privacy protection e.g. health care information, IPR (intellectual property rights) or digital copyrights protection are goals of modern access control.

The UCON_{ABC} utilizes subject attributes and object attributes in its authorization based decision process and includes authorizations, obligations and conditions. Authorizations evaluate subject attributes, object attributes and requested rights together with a set of rules for usage decision. Obligations are requirements that have to be fulfilled for usage allowance. Conditions are environmental or system requirements that are independent from individual subjects. The additional attributes represent properties or capabilities and can be time dependent (before, during or after a usage exercise). Furthermore UCON provides server side as well as client side control architecture.

The UCON_{ABC} model consists of eight components: subjects, subject attributes, objects, object attributes, rights, authorizations, obligations and conditions.

The UCON_{ABC} model is based on a classification of three criteria:

- decision factors (authorizations, obligations, conditions)
- continuity of decision (pre or ongoing with respect to the access in question)
- mutability (allows updates on subject or object attributes in different times (before, during or after))

Correlating this criteria enable the possibility for time dependent actions, checks and changes of conditions in different ways.

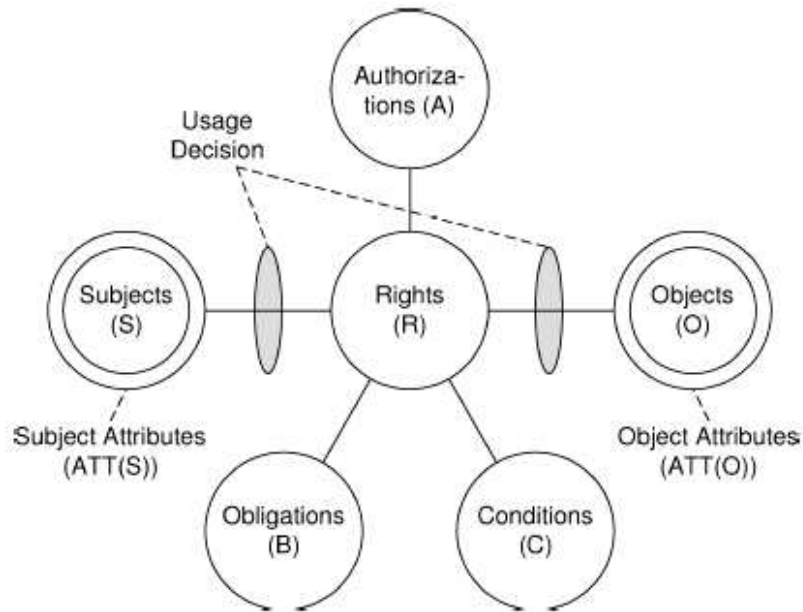


Figure 22: UCON_{ABC} model components

Analysis of Properties:

CR-1 *Decentralized Administration*: Decentralized Administration is supported. UCON integrates traditional and modern access control. RBAC (role based access control) i.a. is covered by UCON. Thus decentralized administration is a main issue of UCON.

CR-2 *Check against Security Principles*: Check against Security Principles is supported. By deployment of attributes and conditions it is possible to define fine grained constraints and preconditions. Attributes can be differenced in immutable and mutable ones. The properties of mutable attributes can be modified in time by a client. Mutable attributes are often used at trust management and digital rights management (DRM).

CR-3 *Separation of Administration*: Separation of Administration, in the meaning of functional separation, is supported. The UCON item 'Rights' permits a delegation of rights and rights for administering access. Rights are privileges that a subject can hold and exercise on an object.

CR-4 *Support for Compliance Rules*: Support for Compliance Rules is supported by implementing conditions, obligations and attributes.

CR-5 *Support for Static Constraints*: Support for Static Constraints is supported by deployment of obligations, attributes and conditions. Obligations are functional predicates that verify mandatory requirements a subject has to perform before or during a usage exercise. Conditions are environmental or system oriented decision factors. Attributes are properties or capabilities.

CR-6 *Monitoring Administrative Operations*: Monitoring Administrative Operations is not within the scope of the UCON model. Administration issues are even not inspected. This core

requirement can be seen as an addition to the UCON model.

CR-7 *Enforce Administrative Access Rights*: Enforce Administrative Access Rights is not within the scope of the UCON model. This core requirement can be seen as an addition to the UCON model.

CR-8 *Distributed Components / Secure Communication*: Distribute Components / Secure Communication are supported. To cover modern access control models like trust management and DRM secure communication is a very important item of the UCON model.

CR-9 *Support Context Information*: Support Context Information is supported by subject and object attributes. An important item is the time dimension for checking constraints and conditions. It is possible to check preconditions or constraints during the runtime of an operation or process.

CR-10 *Execute Consistency and Property Checks*: Execute Consistency and Property Checks are supported by implementing the defined consistency rules of the model.

CR-11 *Policy Management / Use Cases*: Policy Management / Use Cases are not within the scope of the UCON model. This core requirement can be seen as an addition to the UCON model.

CR-12 *Workflows*: Workflows are not within the scope of the UCON model. This core requirement can be seen as an addition to the UCON model.

CR-13 *Special Features / Remarks*: UCON_{ABC} combines traditional access control models with modern and digital access controls. This revision affords a practicable approach of the requirements of modern digital environments. Nevertheless it is difficult to adopt this still theoretical model to all practical processes of a modern enterprise. More practicable examples and an additional supporting tool are desirable.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕⊖
CR-7	Enforce Administrative Access Rights	⊕⊖
CR-8	Distributed Components / Secure Communication	⊕
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕⊖
CR-12	Workflows	⊕⊖
CR-13	Special Features / Remarks	⊕⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.19 USE - UML-based Specification Environment and Object Constraint Language

Selected References:

- Martin Gogolla et al.: USE: A UML-Based Specification Environment for Validating UML and OCL [GBR05].
- Martin Gogolla: Model Development in the UML-based Specification Environment (USE) [Gog07].

Description:

The tool USE is an UML-based specification environment to assist developers in model-driven techniques. USE is basically an interpreter for UML and OCL constraints. Such constraints are either an invariant (i.e. a static constraint) or a pre- or postcondition that can be used to evaluate contextual information at runtime. The USE application is basically a research and teaching utility.

USE applies model checking methodologies by exploring properties of models. The USE system analyzes the model structure and the model behavior (i.e. operations or pre- and postconditions). It is possible to formally check constraints against expectations and derive formal model properties. Figure 23 shows the model definition interface of USE. On the left side the static states are defined as class definitions. The center shows state changes in terms of sequence diagrams. In the bottom right corner OCL expressions and their evaluation results are depicted.

The following description is taken from [GBR05]:

Class diagram: A UML model is given to USE in textual form. The central parts of the example model are shown in the class diagram: One class with attributes and operations, one association with role names and multiplicities, and two enumerations. Structural restrictions (invariants) which determine the allowed object diagrams and behavioral restrictions (pre- and postconditions) which narrow the allowed operation calls are present in the model browser, but are not shown in the class diagram.

Sequence Diagram: Execution of operations is indicated in the sequence diagram. The operation sequence including operation parameters is given to USE in a command shell. In the current operation sequence shown in the screen shot, all pre- and postconditions are satisfied. Failing pre- or postconditions would have been highlighted and could be analyzed in more detail on the command shell. As an example for a complete operation description with pre- and postconditions we show the operation marry.

OCL Expression Evaluation: USE makes it possible to query the current system state by evaluating OCL expressions in an ad-hoc manner. The example query retrieves the name and civil status of alive persons. In SQL, this query would be stated as `select name, civstat from Person where alive=true`. Thus, SQL-like query formulation and evaluation is possible in USE. The class extent window and the evaluate OCL expression window are two ways to inspect the current system state. The developer has the freedom to choose the most appropriate one.

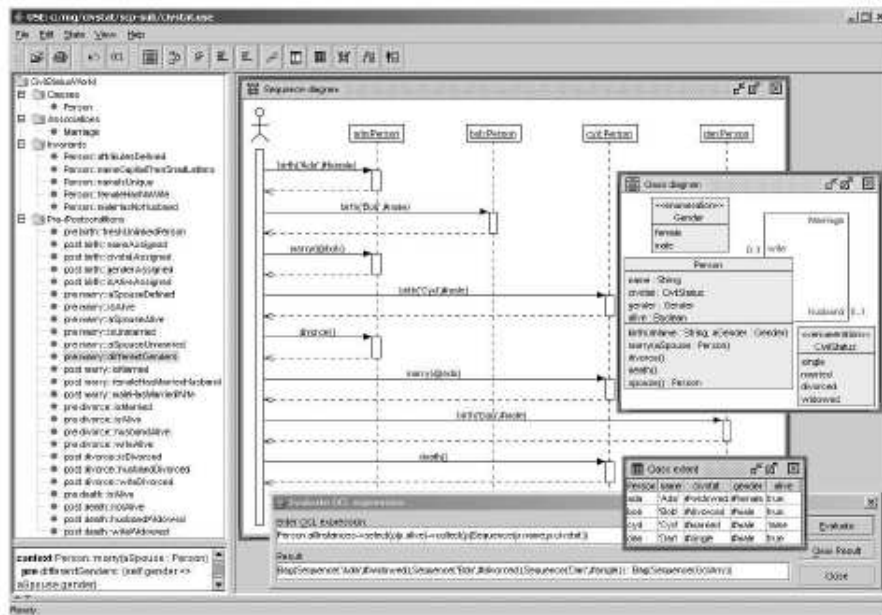


Figure 23: Class Diagram, Sequence Diagram and OCL Evaluation [GBR05]

Analysis of Properties:

CR-1 *Decentralized Administration*: The available documentation does not provide any information about a decentralized model specification. Therefore, we assume a centralized application with a single central model repository.

CR-2 *Check against Security Principles*: Based on USE's pre- and postconditions it is feasible to describe corporate security principles as additional conditions that have to be fulfilled in advance or afterwards (i.e. in terms of postconditions) to perform a valid state transition.

CR-3 *Separation of Administration*: The administration interface should be considered as a research prototype and does not address the specification of administration domains.

CR-4 *Support for Compliance Rules*: Based on USE's pre- and postconditions it is feasible to describe compliance rules as additional conditions that have to be fulfilled in advance before a state transition may occur.

CR-5 *Support for Static Constraints*: The definition of pre- and post conditions and valid system states can be used to describe static constraints for runtime enforcement. Every model violation will be detected at runtime and communicated to a logging instance.

CR-6 *Monitoring Administrative Operations*: USE provides a logging protocol that monitors model constraint violations. In addition, every state change event is mediated to a monitoring instance. Therefore, if the administration process itself is modeled as a sequence diagram, it should be possible to monitor administrative operations. Nevertheless, this seems quite an amount of work.

CR-7 *Enforce Administrative Access Rights*: The USE application does not provide any access control mechanisms for the administration of system model.

CR-8 *Distributed Components / Secure Communication*: The USE application is considered a research prototype that focuses on the expression of complex models and model sim-

ulation. Based on a centralized architecture no distributed or secured communication between components take place.

CR-9 *Support Context Information*: USE is capable of defining pre- and postconditions. These can be used to express contextual information that will be queried at runtime or during a simulation to support dynamic constraints, such as local time or a subject's location.

CR-10 *Execute Consistency and Property Checks*: Similar to Alloy, USE helps to find models with desired properties, such as to query security properties and execute consistency analysis. This is done by testing formal descriptions by simulated system states and state changes. The latter can be considered as simulation scenarios comparable to sequence diagrams. This also allows to identify unwanted properties, such as permission conflicts.

CR-11 *Policy Management / Use Cases*: While USE's primary intention is to define system models, such as set of security policies, it is possible to enrich the system model by the Action Semantics language of UML to define for instance a permission-role assignment function. The screen shot of Figure 24 shows information about a system state generated with the USE tool. Nevertheless, it seems quite complex to define elaborate access control policies or role descriptions (cf. Figure 25).

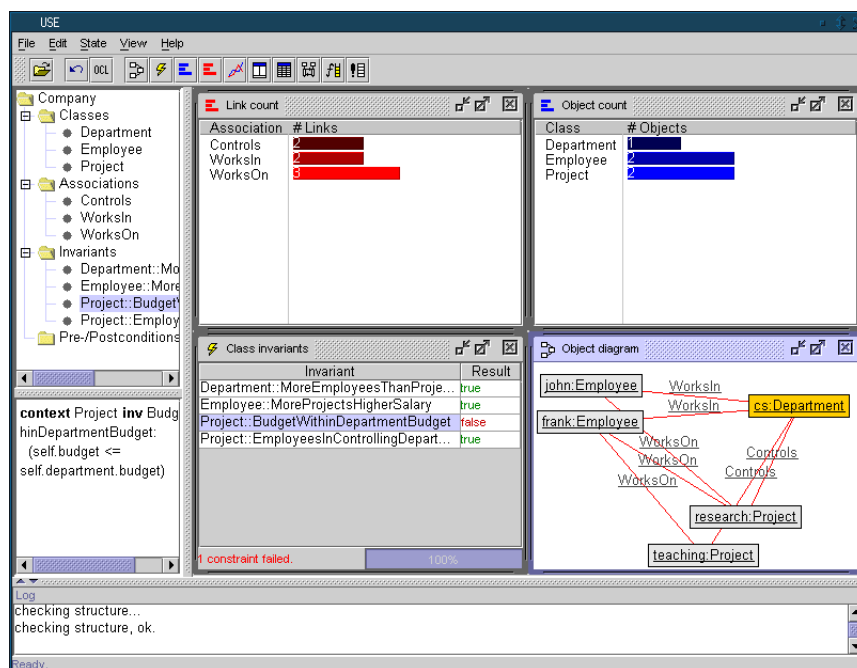


Figure 24: USE Example System State [GBR05]

```

procedure Person_marry(self:Person,aSpouse:Person)
begin
  [self].civstat:=[#married]; [aSpouse].civstat:=[#married];
  if [self.gender=#female] then
    begin Insert(Marriage,[self],[aSpouse]); end
  else -- [self.gender=#male]
    begin Insert(Marriage,[aSpouse],[self]); end;
end;

```

Figure 25: USE Example Role *Marry* [GBR05]

CR-12 *Workflows*: USE can be used to simulate so called scenarios that are comparable with an UML sequence diagram. Typically activity diagrams are the preferred choice to model workflows with UML, but it seems possible to define at least small workflows within USE by utilizing sequence diagrams.

CR-13 *Special Features / Remarks*: The complete USE application framework is still under development with a primary field of engagement in the area of universities. In addition, the whole USE application is considered as a research prototype with no warranty of any kind and a completely centralized architecture.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊖
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊖
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊖
CR-7	Enforce Administrative Access Rights	⊖
CR-8	Distributed Components / Secure Communication	⊖
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕⊖
CR-12	Workflows	⊕⊖
CR-13	Special Features / Remarks	⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.20 eXtensible Access Control Markup Language

Selected References:

- Sun Microsystems: Sun's XACML Implementation Programmer's Guide [Mic04].
- Erik Rissanen and Babak Sadighi Firozabadi: Access-Control Policy Administration in XACML [RF05].
- Anne Anderson: XACML Profile for Role Based Access Control (RBAC) [And04].
- Tim Moses: eXtensible Access Control Markup Language (XACML) Version 2.0 [Mos05].

Description:

XACML is an OASIS standard that describes both a policy language and an access control decision request/response language (both encoded in XML). The policy language is used to describe general access control requirements, and has standard extension points for defining new functions, data types, combining logic, etc. The request/response language lets you form a query to ask whether or not a given action should be allowed, and interpret the result. The response always includes an answer about whether the request should be allowed using one of four values: Permit, Deny, Indeterminate (i.e. an error occurred), or Not Applicable (i.e. the request cannot be answered).

The model operates by the following steps (taken from [Mos05]):

1. Policy administration points (PAP) write policies and policy sets and make them available to the PDP. These policies or policy sets represent the complete policy for a specified target.
2. The access requester sends a request for access to the PEP.
3. The PEP sends the request for access to the context handler in its native request format, optionally including attributes of the subjects, resource, action and environment.
4. The context handler constructs an XACML request context and sends it to the PDP.
5. The PDP requests any additional subject, resource, action and environment attributes from the context handler.
6. The context handler requests the attributes from a policy information point (PIP).
7. The PIP obtains the requested attributes.
8. The PIP returns the requested attributes to the context handler.
9. Optionally, the context handler includes the resource in the context.
10. The context handler sends the requested attributes and (optionally) the resource to the PDP.
11. The PDP evaluates the policy.
12. The PDP returns the response context (including the authorization decision) to the context handler.

13. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP.
14. The PEP fulfills the obligations.
15. If access is permitted, then the PEP permits access to the resource; otherwise, it denies access.

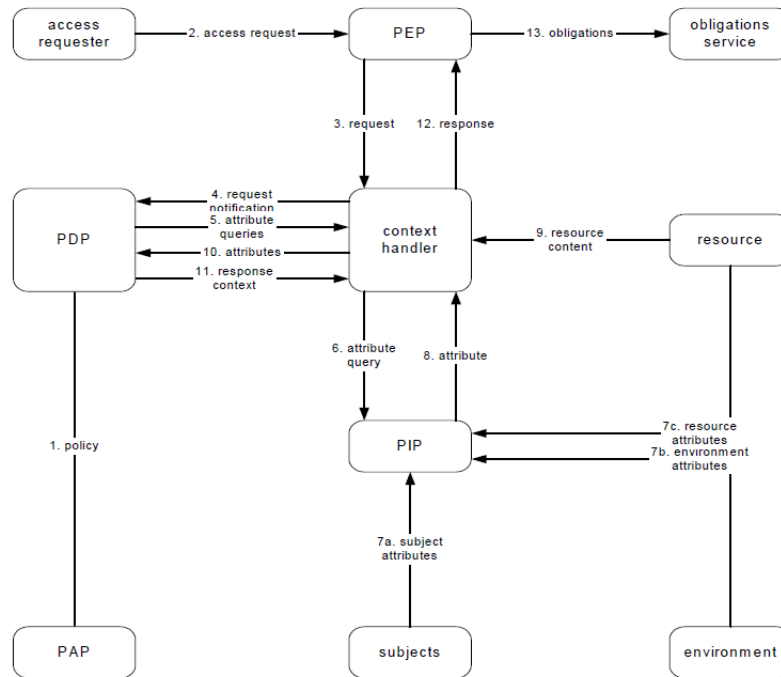


Figure 26: XACML Data Flow Diagram [Mos05]

Analysis of Properties:

- CR-1 Decentralized Administration:** XACML supports distributed policy administration. This means that a composite policy may refer to policies kept in arbitrary locations. The result is that rather than having to manage a single monolithic policy, different people or groups can manage separate sub-policies as appropriate, and XACML knows how to correctly combine the results from these different policies into one decision.
- CR-2 Check against Security Principles:** XACML policies are based on the comparison of subject attributes and policy attributes for accessing resources. Corporate security policies, such as signing every exchanged message, can not be expressed with XACML because policies are bound to concrete actions and can not be used to express general security statements.
- CR-3 Separation of Administration:** Due to its network tailored specification, XACML supports distributed policy repositories. Thus, it is possible to define dedicated policy repositories, each containing policies belonging to a specific domain.
- CR-4 Support for Compliance Rules:** XACML is designed as an XML-based specification language for expressing and enforcing access control policies. There exists no mechanism

to integrate compliance rules, but it seems feasible to extend an existing XACML implementation to access business compliance rules such as JRules.

- CR-5 *Support for Static Constraints*: XACML is a general purpose policy system, designed to support the needs of most authorization systems. Therefore, it directly supports static security constraints as well as dynamic constraints based on contextual or situational environment information.
- CR-6 *Monitoring Administrative Operations*: The XACML specification does not contain recommendations about monitoring administrative or user operations. This is part of a policy enforcement point or a policy decision point. In addition, as stated for Core Requirement 4, it is not possible to express compliance rules, such as access request auditing within XACML policies.
- CR-7 *Enforce Administrative Access Rights*: In [RF05] an extension for XACML is proposed that allows to specify security policies for administrative tasks. Meaning, it is possible to express restrictions in terms of delegations. Adding delegation to XACML involves defining policies that can express administrative rights, and a new processing model that can verify that delegations have been performed in an authorized manner.
- CR-8 *Distributed Components / Secure Communication*: XACML itself only provides a specification language to express access control policies. To enable the distribution of policies and the secure message exchange additional standards have to be integrated on top of XACML, such as SAML for attribute exchange and WS-Security for message security.
- CR-9 *Support Context Information*: XACML defines attributes to evaluate contextual information. Attributes are named values of known types that may include an issuer identifier or an issue date and time. Specifically, attributes are characteristics of the subject, resource, action, or environment in which the access request is made. For example, A user's name, their group membership, a file they want to access, and the time of day are all attribute values. When a request is sent attribute values of the request are compared with the attribute values defined in a policy to make the access decisions [Mic04].
- CR-10 *Execute Consistency and Property Checks*: XACML supports schema validation when a request is handled. While some structural errors will be caught by the parsing routines, there is no semantic evaluation defined in the XACML specification. Nevertheless, XACML defines so called attribute selectors that perform XQueries. Therefore, the necessary functionality is already provided by Sun's XACML implementation a validation and consistency analyzer must be implemented.
- CR-11 *Policy Management / Use Cases*: Based on the proposed XACML extension for role-based access control [And04]. It is possible to define policies and roles. In addition, policies can be assigned to one or more roles and users. XACML policies only reference subjects. Subjects must be maintained in an external organizational repository, such as an LDAP directory. Further, there exists no validation against an organizational repository if a specified subject does exist in the repository or not.
- CR-12 *Workflows*: Control-flow analysis is not part of the XACML specification. While it is possible to define policies for dedicated actions that can be considered as tasks, strict least privilege has to be expressed explicitly within a policy (e.g. Task 2 may only be accessed after Task1 was accessed and completed). A direct integration of workflow models, for

instance as XPDL is not part of the XACML specification. Therefore, every change in the related workflow model could result in inconsistent or invalid XACML policies.

CR-13 *Special Features / Remarks*: The logic within a policy uses an extensible system of data types and functions to promote interoperability. All attributes used in XACML are of a well-known type, and all functions have well known signatures that use these same data types.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊖
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕⊖
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊖
CR-7	Enforce Administrative Access Rights	⊕
CR-8	Distributed Components / Secure Communication	⊕⊖
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊕⊖
CR-11	Policy Management / Use Cases	⊕⊖
CR-12	Workflows	⊕⊖
CR-13	Special Features / Remarks	⊕⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.21 Temporal role-based access control model (TRBAC, GTRBAC, X-GTRBAC)

Selected References:

- Elisa Bertino, Piero Andrea Bonatti, Elena Ferrari: TRBAC: A Temporal Role-based Access Control Model; University of Milano.[BBF00b]
- Elisa Bertino, Piero Andrea Bonatti, Elena Ferrari: A Temporal Role-Based Access Control Model; University of Milano, University of Como, August 2001.[BBF00a]
- Rafae Bhatti, Arif Ghaffor, Elisa Bertino, James B.D. Joshi: X-GTRBAC: An XML-Based Policy Specification Framework and Architecture for Enterprise-Wide Access Control; University of Pittsburgh, Purdue University, West Lafayette, 2005.[BGBJ05]
- Rafae Bhatti, James B.D. Joshi, Elisa Bertino, Arif Ghafoor: X-GTRBAC Admin: A Decentralized Administration Model for Enterprise Wide Access Control; University of Pittsburgh, Purdue University, West Lafayette, 2004.[BSB⁺05]
- James B.D. Joshi, Elisa Bertino, Usman Latif, Arif Ghafoor: A Generalized Temporal Role-Based Access Control Model; University of Pittsburgh, Purdue University, West Lafayette, 18. November 2004.[JBLG05]
- James B. D. Joshi, Elisa Bertino, Basit Shafiq, Arif Ghafoor: Dependencies and Separation of Duty Constraints in GTRBAC; Purdue University, West Lafayette, University of Milano, 2003.[JBSG03]
- James B D Joshi and Elisa Bertino: Fine-grained Role-based Delegation in Presence of the Hybrid Role Hierarchy; University of Pittsburgh, Purdue University, West Lafayette, 2006.[JB06]

Model description:

Modern information sciences try to couple theoretical access control models with requirements of dynamic modern enterprises. A wide range of security policies, e.g. discretionary and mandatory policies as well as user-defined or organizational specific policies and workflow-based systems shall be covered by Temporal Role-Based Access Control (TRBAC).

The TRBAC-model-group (with GTRBAC and X-GTRBAC) starts the examination to combine the theoretical requirements with dynamical ones. TRBAC is an extension of RBAC (role-based access control). It supports periodic activations and deactivations of roles and temporal dependencies among such actions expressed by means of role triggers. To express temporal constraints the model uses active rules, trigger priority, deferred actions and run-time requests. The system is composed by six main modules:

- trigger support
- periodic event handler
- run-time request handler
- deferred action handler

- action handler
- safeness checker

In addition to the pure RBAC model enhanced models like 'Generalized Temporal Role-Based Access Control Model' (GTRBAC) and X-GTRBAC for XML-based systems are developed. Furthermore the issue 'Administration' is analysed in the model of X-GTRBAC Admin.

GTRBAC provides duration and periodicity constraints as well as other forms of specialized activation constraints. Key aspect is the distinction between notions of role enabling and role activation. Thus a role can possess the states 'disabled', 'enabled' and 'active'. GTRBAC offers a wide range of temporal constraints like duration constraints on roles, on user-role assignment and on role-permission assignment. Furthermore GTRBAC includes cardinality constraints, maximum active duration constraints and allows expressing role hierarchies and separation of duty. The model offers the following types of constraints:

- temporal constraints
 - role enabling
 - user-role assignment
 - role-permission assignment
 - role activation
- run-time events
- triggers

X-GTRBAC is a combination of the temporal framework of GTRBAC and an attribute based constraint specification. A XML-based specification language is developed to cover the functional requirements like RBAC and to provide a syntactic and semantic construct to enforce temporal constraints of GTRBAC. The required policies are also covered by this model.

X-GTRBAC Admin possesses in addition the administration component. It allows decentralization of policy administration, specifying the domain of authority of the system administrators and provides mechanism to distribute the administrative authority over multiple domains within enterprises.

Analysis of Properties:

CR-1 *Decentralized Administration:* Decentralized Administration is supported by the model of X-GTRBAC Admin. Distributed locations are covered by a XML-interface which permits a spreading of administration and authorisation.

CR-2 *Check against Security Principles:* Check against Security Principles is supported by the implemented different constraints like periodicity and duration constraints on role enabling and role activation. With X-GTRBAC the possibility for policy definition and administration is possessed.

- CR-3 *Separation of Administration*: Separation of Administration, in the meaning of functional separation, is supported by the model of X-GTRBAC Admin. Decentralization and delegation are main items of this model. With respect to modern dynamic enterprise structures, separate administration must be possible. X-GTRBAC Admin encourages decentralization of administration tasks and separation of duty.
- CR-4 *Support for Compliance Rules*: Support for Compliance Rules is supported by duration and periodicity constraints. The combination of RBAC with dynamic and temporal constraints allows an adaption of enterprise rules and conditions.
- CR-5 *Support for Static Constraints*: Support for Static Constraints is supported. GTRBAC offers some different constraints. A set of dynamic constraints are also offered by the model. Thus time dependend actions and operations can also be handeled via GTRBAC.
- CR-6 *Monitoring Administrative Operations*: Monitoring Administrative Operations are supported. Administrative aspects are discussed in X-GTRBAC Admin. All proposals are approved via a mathematical model.
- CR-7 *Enforce Administrative Access Rights*: Enforce Administrative Access Rights is supported by X-GTRBAC Admin.
- CR-8 *Distributed Components / Secure Communication*: Distribute Components / Secure Communication is supported and a main item in GTRBAC. Modern companies operate with distributed locations and components. GTRBAC possesses a proposal concerning these requirements.
- CR-9 *Support Context Information*: Support Context Information is supported by the application of constraints with their contextual information. Important time parameters for dynamic processes are also implemented by GTRBAC.
- CR-10 *Execute Consistency and Property Checks*: Execute Consistency and Property Checks are supported by the verifications of the model. The TRBAC authors make a lot of approvals for checking and verifying their model. and properties of the classified tasks of the model.
- CR-11 *Policy Management / Use Cases*: Policy Management / Use Cases are supported by (X)-GTRBAC. Policies can be defined and administrated by a XML based interface.
- CR-12 *Workflows*: Workflows are featured in the GTRBAC model. By X-GTRBAC it is possible to manage these defined workflows via a XML-interface.
- CR-13 *Special Features / Remarks*: TRBAC with its extensions GTRBAC and X-GTRBAC is a very interesting model. It contains all theoretical and mathematical ideas and verifications of RBAC as well as requirements of modern dynamic enterprises. The model is implemented as a Java application. Thus it is usable in known development environments. Possibly it would be complicated to adapt the theoretical model with its much difficult verifications to the practical ORKA-requirements.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕
CR-7	Enforce Administrative Access Rights	⊕
CR-8	Distributed Components / Secure Communication	⊕
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊕
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊕
CR-13	Special Features / Remarks	⊕

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

3.22 Integrated Approach to Engineer and Enforce Context Constraints in RBAC Environments (xoRBAC)

Selected References:

- Gustaf Neumann and Mark Strembeck: Design and Implementation of a Flexible RBAC-Service in an Object-Oriented Scripting Language [NS01].
- Gustaf Neumann and Mark Strembeck: An Integrated Approach to Engineer and Enforce Context Constraints in RBAC Environments. [NS04].
- Gustaf Neumann and Mark Strembeck: A Scenario-driven Role Engineering Process for Functional RBAC Roles. [NS02].
- Mark Strembeck: <http://wi.wu-wien.ac.at/home/mark/xoRBAC/index.html>. [Str].

Description:

The xoRBAC component is a software module providing a flexible access control decision service based on RBAC. It goes together with arbitrary applications on Windows and Unix platforms accessing xoRBAC through a C or TCL interface. In particular, there has been research on deploying xoRBAC in an HTTP environment for a web-based mobile code system with promising results for this field of application. The xoRBAC architecture is illustrated in Figure 27.

While xoRBAC is able to deal with static separation of duty and cardinality constraints for

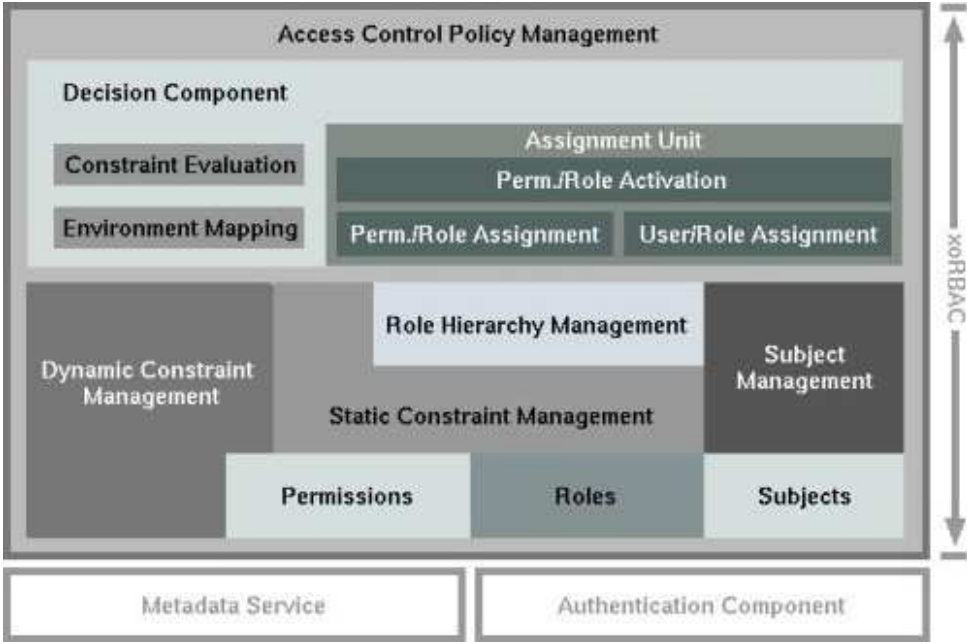


Figure 27: xoRBAC: Conceptual Structure

roles and permissions, one of its major features is the support of context constraints making xoRBAC highly flexible and dynamically extensible. By using context constraints, access control decisions can be based on conditions about arbitrary contextual input (including time) from both software and hardware sensors. Furthermore, it is possible to add context constraints and

even sensors during the runtime.

The policy is stored by the serialization of the runtime objects of xoRBAC using RDF data models in XML syntax. Thus, by not having a dedicated policy language, policy validation by external tools becomes difficult. There is a graphical administration tool, however, which makes it more feasible to read and write the running policy.

Analysis of Properties:

- CR-1 *Decentralized Administration:* The xoRBAC software module is a box containing the policy manager and in particular a policy decision point with an integrated policy repository. Even though xoRBAC provides an API and can be integrated into an HTTP environment, the administration component can not get separated from its policy repository. Only a cascaded communication between several xoRBAC instances is possible but no composition of locally defined policies. Consistency checks of separately administrated policies should be added to realize this core requirement. Permission-role reviews are supported, especially in distributed systems.
- CR-2 *Check against Security Principles:* Check against Security Principles is supported by using constraints and context information. Context constraints can be added and evaluated even at runtime.
- CR-3 *Separation of Administration:* Separation of Administration is supported by defining static separation of duty constraints and maximum and minimum cardinalities for both roles and permissions. These features can be used to differentiate between user and administrator rights and to limit the permissions of a certain administrator.
- CR-4 *Support for Compliance Rules:* Support for Compliance Rules is supported by using constraints and context information to restrict administrator and user rights. Further more xoRBAC is associated with a meta data service that captures logging and auditing information. Thereby compliant auditing is performed.
- CR-5 *Support for Static Constraints:* xoRBAC is able to deal with static separation of duty and cardinality constraints for roles and permissions. Its Role Hierarchy Management uses the static constraint management component to prevent the creation of role hierarchies that are disallowed by the SSD constraints or cardinalities within the system.
- CR-6 *Monitoring Administrative Operations:* xoRBAC is associated with a meta data service that captures logging and auditing information. It is not brought up whether this service is performed in a secure manner.
- CR-7 *Enforce Administrative Access Rights:* xoRBAC deals with work files where all tasks a certain type of user can perform are described. This is done in accordance with the constraint catalog and is used for a semi-automatic creation of a preliminary role-hierarchy. Further more xoRBAC is associated with a authentication service. Thereby it should be possible to check and restrict any access operation after the initial concrete RBAC model has been designed.
- CR-8 *Distributed Components / Secure Communication:* According to the authentication service an encryption service should be added as a associated service to the xoRBAC software module.

CR-9 *Support Context Information*: One of the major features of xoRBAC is the support of context constraints. By using context constraints, access control decisions can be based on conditions about arbitrary contextual input (including time) from both software and hardware sensors. Furthermore, it is possible to add context constraints and even sensors during the runtime.

CR-10 *Execute Consistency and Property Checks*: xoRBAC provides extensive review functions by means of its graphical administration tool but does not provide consistency and property checks. The policy is stored using RDF data models in XML syntax but no dedicated policy language is used.

CR-11 *Policy Management / Use Cases*: xoRBAC can be administered by using either its application programming interface (API) or a tailored graphical tool that supports the complete set of functions offered by xoRBAC. Its main features are

- Many-to-many user-role and permission-role assignment (and revocation).
- Arbitrary (DAG) role-hierarchies (permission-inheritance).
- Definition of conditional permissions via context constraints.
- Static separation of duties constraints for both roles and permissions (SSD constraint-inheritance via the role-hierarchy).
- Maximum and minimum cardinalities for both roles and permissions.
- Extensive review functions (introspection), e.g. subject-role review, permission-role review, subject-permission review.
- Serialization (import and export) of xoRBAC policies as RDF metadata in XML Syntax.

CR-12 *Workflows*: It should be possible to define workflows by means of the scenario-driven role engineering process for functional RBAC roles. This method is a serialization of the runtime objects of xoRBAC using RDF data models in XML syntax. It describes all tasks a certain type of user can perform in accordance with the catalog of all defined constraints. The description is based on a business process model where each action and event is associated with a particular access operation.

CR-13 *Special Features / Remarks*: The characteristic features of xoRBAC are the handling of context information and the scenario driven engineering approach. Both are already described below.

Evaluation of Core Requirements:

CR	Description of CR	Valuation
CR-1	Decentralized Administration	⊕⊖
CR-2	Check against Security Principles	⊕
CR-3	Separation of Administration	⊕
CR-4	Support for Compliance Rules	⊕
CR-5	Support for Static Constraints	⊕
CR-6	Monitoring Administrative Operations	⊕
CR-7	Enforce Administrative Access Rights	⊕
CR-8	Distributed Components / Secure Communication	⊕⊖
CR-9	Support Context Information	⊕
CR-10	Execute Consistency and Property Checks	⊖
CR-11	Policy Management / Use Cases	⊕
CR-12	Workflows	⊕
CR-13	Special Features / Remarks	⊕⊖

⊕: supported, ⊕⊖: may be supported with some effort, ⊖: not supported

4 Recommended Models

In this section we will give an overview of the analyzed models and their advantages for use within the ORKA project.

- Adage
 - focused on the creators of authorization policies for distributed enterprises:
 - * a graphical user interface (GUI) for policy definition and management exists
 - * a textual Authorization Language (AL) for expressing policies exists for power users
 - provides a modular architecture that allows to integrate external security services like SSL
- ARBAC97/99/02
 - important fundamental ideas and verifications about roles based access control and its administration
- C-TMAC
 - special analysis about the distinction of constraints (e.g. separation of duty)
 - integration of context informations and modelling
- Conditional Delegation in Secure Workflows
 - The model is interesting concerning the construction of algorithms for automated consistency checking of constraints in workflows but not in relation to a convenient administration model.
- CoSAWoE
 - implementation of a policy decision point
- Graph-Based Formalism for RBAC
 - theoretical framework, general purpose graph transformation tools like AGG may be used for administration
 - easy to translate it in a suitable form readable by ORKA's validation component
- A Model of OASIS Role-Based Access Control and its Support for Active Security
 - focused on access control in distributed and decentralized systems
 - dynamic aspects of role activation and deactivation characterize the model
 - formal methods are used
- Ponder
 - intensive examination of policies and their executions (solved by an own language)
 - toolkit available

- Role Control Center RCC
 - relatively simple graphic-centered model
 - particularly developed to support authorization administration tasks
 - decentralized administration supported
 - implementation and tools available (suitable for web-based applications)
 - it should be possible to transform the role graphs into a logical form for validation
- Role Graph Model
 - development of the model continues constant
 - important administration model
 - concept of delegation
 - implementation and tools available
 - parts of a language in XML
 - algorithms to combine several role graphs (for combination of several policies)
- USE + UML/OCL
 - MDA-tools available
 - validation tools available
- X-GTRBAC
 - practicable RBAC upgrade (e.g. workflows integrated)
 - administration considered
 - XML interface
- xoRBAC
 - support of context constraints as a main feature
 - based on RBAC, administration considered
 - workflows considered
 - developed for an HTTP environment for a web-based mobile code system

- XACML

This model provides some interesting aspects with respect to the underlying concepts. The combination of different policies from different locations identified by unique resource identifiers to form a complex policy builds the fundament to satisfy the core requirement *CR-1 Decentralized Administration*. The modular approach of XACML policies as well as the available extension profiles enrich the underlying concepts of XACML to support the core requirement *CR7-Enforce Administrative Access Rights* (e.g. XACML delegation profile) and *CR-8 Distributed Components* (e.g. XACML-to-SAML mapping). XACML's condition element enables the specification of history-based expressions in workflow environments, satisfying core requirement *CR-12 Workflows*. It's ability to be

extended by additional profiles and the fact that XACML is an accepted industrial standard (There exists implementations for BEA WebLogic, Securent, JBoss, and IBM WebSphere) gives XACML a plus regarding the special features addressed by core requirement 13.

The major drawback of XACML from an administrative perspective is the lack of supporting administrative tools. While enforcement and decision engines are already in place for XACML, elaborated administration and specification tools are currently not at hand and must be build from scratch within the ORKA project.

Features of these models will be considered in our concept and design of an ORKA administration security model.

5 Conclusion

The discussion in work package 5.1 about requirements for the ORKA project led to a compilation of requirement, split into mandatory and optional requirements. In work package 5.2 we extracted 13 core requirements which we consider essential for the ORKA project. These core requirements have been used to analyze and evaluate different security administration models. Our intention was to identify one or more of these models as a foundation for the architecture of the ORKA administration model.

The analyzed administration models follow very different approaches. Some are scientific, theoretical concepts with mathematical validation and only few approaches for support of existing dynamically enterprise processes. On the other hand there are pragmatial approaches adapting theoretical concepts to modern enterprise requirements. Some models provide special languages and toolkits, thus an adaption of the model to application implementation seems to be possible. As an example, workflows as an appropriate business process tool have been considered in some models.

We analyzed the security administration models with respect to these thirteen core requirements and developed a standardized catalog of evaluation criteria. For each requirement, we checked if and to which extent it is supported by the model:

- \oplus : the requirement is supported by the model
- \ominus : the requirement is not supported by the model
- $\oplus\ominus$: the requirement is not supported currently but may easily be implemented within the ORKA project

After analyzing and evaluating the models we decided to use some aspects of different models and recommend them for the following processes of the ORKA project. Some models are based on interesting ideas and concepts which can be useful to design the ORKA demonstrator. Documentation about the most beneficial and expedient aspects has been collected in section 4 of this work package.

Within this subject area the models *Role Graph Model* and *X-GTRBAC* shall be considered in particular. They offer important aspects for administration of policy management systems. *Role Graph Model* provides an important concept of delegation, a rudimental language in XML,

various practicable tools and an implementation of validation. *X-GTRBAC* particularly provides an useful concept of administration which can support the design of the ORKA administration model. Furthermore it offers an XML interface and important features to implement static and dynamic constraints.

Based of these considerations the ORKA model for the administration of policy management systems will be designed and implemented in work package 5.3. Furthermore we ask for consideration of our analyzing output within the development of the policy language (ORKA subject area SPEC).

References

- [Ahn99] G.-J. Ahn. *The RCL 2000 language for specifying role-based authorization constraints*. PhD thesis, George Mason University, Fairfax, Virginia, 1999.
- [And04] Anne Anderson. Xacml profile for role based access control (rbac), 2004. <http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf>.
- [AS00] Gail-Joon Ahn and Ravi Sandhu. Role-based authorization constraints specification. *ACM Trans. Inf. Syst. Secur.*, 3(4):207–226, 2000.
- [AW05] V. Atluri and J. Warner. Supporting conditional delegation in secure workflow management systems. In *SACMAT*, pages 49–58, 2005.
- [BBF00a] E. Bertino, P.A. Bonatti, and E. Ferrari. TRBAC: A temporal role-based access control model. In *Proc. of the 5th ACM Workshop on Role-Based Access Control*, pages 21–30, N.Y., July 26–27 2000. ACM Press.
- [BBF00b] Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. TRBAC: a temporal role-based access control model. In *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*, pages 21–30, New York, NY, USA, 2000. ACM Press.
- [BFA99] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security (TISSEC)*, 2(1):65–104, 1999.
- [BGBJ05] R. Bhatti, A. Ghafoor, E. Bertino, and J. Joshi. X-GTRBAC: an XML-based policy specification framework and architecture for enterprise-wide access control. *ACM Transactions on Information and System Security*, 8(2):187–227, 2005.
- [Bot01] Reinhardt A Botha. CoSAWoE - A Model for Context-Sensitive Access Control in Workflow Environments, 2001.
- [BSB⁺05] Rafae Bhatti, Basit Shafiq, Elisa Bertino, Arif Ghafoor, and James B. D. Joshi. X-grbac admin: A decentralized administration model for enterprise-wide access control. *ACM Trans. Inf. Syst. Secur.*, 8(4):388–423, 2005.
- [Dam02] N. Damianou. A policy framework for management of distributed systems, 2002.
- [DDLS01a] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. *Lecture Notes in Computer Science*, 1995:18–??, 2001.
- [DDLS01b] Naranker Dulay, Nicodemos Damianou, Emil Lupu, and Morris Sloman. A policy language for the management of distributed agents. In *AOSE*, pages 84–100, 2001.
- [EBM06] D. M. Eyers, J. Bacon, and K. Moody. OASIS role-based access control for electronic health records. *IEE Proceedings Software*, Volume 153(Number 1):Pages 16–23, February 2006.
- [EEKR99] H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages and Tools*. World Scientific, Singapore, 1999.

- [FACG03] D. F. Ferraiolo, G-J. Ahn, R. Chandramouli, and S. I. Gavrila. The Role Control Center: Features and Case Studies. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies*, pages 12–20. ACM, 2003.
- [FKC03] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli. *Role-Based Access Control*. Artech House, 2003.
- [GBR05] Martin Gogolla, Fabian Buettner, and Mark Richters, 2005.
- [GMNP02] C.K. Georgiadis, I. Mavridis, G. Nikolakopoulou, and G. Pangalos. Implementing context and team based access control in healthcare intranets. *Medical informatics and the internet in medicine*, 27(3):185–201, 2002.
- [GMP00] C. Georgiadis, I. Mavridis, and G. Pangalos. Context and Role Based Hybrid Access Control for Collaborative Environments. In *Proceedings of the Fifth Nordic Workshop on Secure IT Systems - Encouraging Co-operation (NORDSEC 2000)*, Reykjavik, Iceland, October 2000.
- [GMPT01] C.K. Georgiadis, I. Mavridis, G. Pangalos, and R.K. Thomas. Flexible team-based access control using contexts. In *Proc. of the ACM Symposium on Access Control Models and Technologies*, pages 21–27, May 3–4 2001.
- [GO04] Mei Ge and Sylvia Osborn. A Design for Parameterized Roles. In P. Samarati and C. Farkas, editors, *Data and Applications Security XVIII – Status and Prospects*. Kluwer, 2004.
- [Gog07] Martin Gogolla. Model Development in the UML-based Specification Environment (USE). *Dagstuhl Seminar Proceedings 06351*, 2007.
- [HO98] Lingling Hua and Sylvia Osborn. Modeling UNIX Access Control with Role Graph. In *Proceedings of International Conference on Computers and Information*, June 1998.
- [IO02] C.M. Ionita and S. Osborn. Privilege Administration for the Role Graph Model. In *Proc. IFIP WG11.3 Working Conference on Database Security*, July 2002.
- [JB06] James B D Joshi and Elisa Bertino. Fine-grained Role-based Delegation in Presence of the Hybrid Role Hierarchy. In *Proc. of the ACM Symposium on Access Control Models and Technologies*, pages 81–90. ACM Press, 2006.
- [JBLG05] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1):4–23, 2005.
- [JBSG03] James B D Joshi, Elisa Bertino, Basit Shafiq, and Arif Ghafoor. Dependencies and Separation of Duty Constraints in GTRBAC. In *Proc. of the ACM Symposium on Access Control Models and Technologies*, pages 51–64. ACM Press, 2003.
- [KMPP02] M. Koch, L. V. Mancini, and F. Parisi-Presicce. A Graph Based Formalism for RBAC. *ACM Transactions on Information and System Security (TISSEC)*, 5(3):332–365, August 2002.
- [Mic04] Sun Microsystems. Sun’s xacml implementation programmer’s guide, 2004. <http://sunxacml.sourceforge.net/guide.html>.
- [Mos05] Tim Moses. extensible access control markup language (xacml) version 2.0, 2005.

- [N.N02] N.N. The ponder policy based management toolkit, 2002.
- [NO94] M. Nyanchama and S. Osborn. Access Rights Administration in Role-Based Security Systems. In Biskup, Morgenstern, and Landwehr, editors, *Database Security VIII: Status and Prospects*, pages 37–56, 1994.
- [NO95] Matunda Nyanchama and Sylvia Osborn. The Role Graph Model. In *First ACM Workshop on Role-Based Access Control*, Gaithersburg Maryland, November 1995.
- [NO99] Matunda Nyanchama and Sylvia Osborn. The role graph model and conflict of interest. *ACM Transaction on Information and System Security*, 2(1):3–33, 1999.
- [NS01] Gustaf Neumann and Mark Strembeck. Design and Implementation of a Flexible RBAC-Service in an Object-Oriented Scripting Language. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*. ACM Press, 2001.
- [NS02] Gustaf Neumann and Mark Strembeck. A Scenario-driven Role Engineering Process for Functional RBAC Roles. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM Press, 2002.
- [NS04] Gustaf Neumann and Mark Strembeck. An Integrated Approach to Engineer and Enforce Context Constraints in RBAC Environments. *ACM Transactions on Information and System Security*, 7(3):392–427, 2004.
- [OG00] S. Osborn and Y. Guo. Modeling Users in Role-Based Access Control. In *Fifth ACM Workshop on Role-Based Access Control*, Berlin, July 2000.
- [OHL03] S. Osborn, Y. Han, and J. Liu. A methodology for managing roles in legacy systems. In *Proceedings of the 8th ACM Symposium on Access control models and technologies (SACMAT)*, pages 33–40. ACM, Jun 2003.
- [Osb02a] Sylvia Osborn. Information Flow Analysis of an RBAC System. In *Proceedings of the ACM SACMAT*, pages 163–168, June 2002.
- [Osb02b] Sylvia Osborn. Integrating Role Graphs: A Tool for Security Integration. *Data Knowl. Eng.*, 43(3):317–333, 2002.
- [PBE01] Stephen Perelson, Reinhardt Botha, and Jan Eloff. Soda: A model for the administration of separation of duty requirements in workflow systems, 2001.
- [PS04] Jaehong Park and Ravi Sandhu. The uconabc usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, 2004.
- [RF05] Erik Rissanen and Babak Sadighi Firozabadi. Access-control policy administration in xacml, 2005. http://www.ercim.org/publication/Ercim_News/enw63/sadighi.html.
- [RSSS94] R. Ramakrishnan, D. Srivastava, S. Sudarshan, and P. Seshadri. The coral deductive system. *The International Journal on Very Large Data Bases (VLDB)*, 3(2):161–210, 1994.

- [SBM99] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 Model for Role-Based Administration of Roles. *ACM Transactions on Information and System Security*, 2(1):105–135, 1999.
- [Sch04] Andreas Schaad. An extended analysis of delegating obligations. In *DBSec*, pages 49–64, 2004.
- [SM99] Ravi S. Sandhu and Qamar Munawer. The arbac99 model for administration of roles. In *ACSAC*, pages 229–, 1999.
- [SM02] Andreas Schaad and Jonathan D. Moffett. A framework for organisational control principles. In *ACSAC*, pages 229–238, 2002.
- [SO02] Seog Park Sejong Oh. Task - role-based access control model, 2002.
- [SO06a] Xinwen Zhang Sejong Oh, Ravi Sandhu. An Effective Role Administration Model Using Organization Structure. *ACM Transactions on Information and System Security*, 9(2):113–137, 2006.
- [SO06b] Yunyu Song and Sylvia Osborn. Conflict of Interest in the Administrative Role Graph Model. In *Secure Data Management 2006*, pages 100–114, 2006.
- [Str] Mark Strembeck. <http://wi.wu-wien.ac.at/home/mark/xoRBAC/index.html> (2006-10-16).
- [Sys06a] BEA Systems. Beaweblogic server: Introduction to weblogic security, 2006. <http://edocs.bea.com/wls/docs81/pdf/secintro.pdf>.
- [Sys06b] BEA Systems. Understanding weblogic resource security, 2006. <http://edocs.bea.com/wls/docs92/secwres/understdg.html>.
- [Sys06c] BEA Systems. Using xacml documents to secure weblogic resources, 2006. <http://edocs.bea.com/wls/docs92/secwres/xacmlusing.html>.
- [Tho97] Roshan K. Thomas. Team-based access control (tmac): a primitive for applying role-based access controls in collaborative environments. In *ACM Workshop on Role-Based Access Control*, pages 13–19, 1997.
- [WO03] H. Wang and S. Osborn. An Administrative Model for Role Graphs. In *Proc. IFIP WG11.3 Working Conference on Database Security*, 2003.
- [WO04] J. Wang and S. Osborn. A Role-Based Approach to Access Control for XML Databases. In *Proceedings of the SACMAT 2004*, June 2004.
- [WO06] He Wang and Sylvia Osborn. Delegation in the role graph model. In *Proceedings of the SACMAT 2006*, 2006.
- [WS03] Ruben Wolf and Markus Schneider. Context-dependent Access Control for Web-based Collaboration Environments with Role-based Approach. In *Second International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS 2003)*, volume 2776 of *LNCS*. Springer, 2003.
- [WSK03] Ruben Wolf, Markus Schneider, and Thomas Keinz. A Model for Context-dependent Access Control for Web-based Services with Role-based Approach.

In *DEXA Intern. Workshop on Network-Based Information Systems, NBIS 2003*, pages 209–214, Prague, Sept. 2003. IEEE Computer Society Press.

- [YMB01] W. Yao, K. Moody, and J. Bacon. A model of OASIS role-based access control and its support for active security. In *SACMAT*, pages 171–181, 2001.
- [ZLZ⁺06] Li Zhang, Lili Luo, Liyong Zhang, Tiesuo Geng, and Zongge Yue. Task-role-based access control in application on mis. 2006.
- [ZSS99] M.E. Zurko, R. Simon, and T. Sanfilippo. A user-centered, modular authorization service built on an RBAC foundation. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 57–71, Oakland, CA, May 1999. IEEE Computer Society Press.