FORBAC: A Flexible Organisation and Role-Based Access Control Model for Secure Information Systems

Oumaima Saidani* and Selmin Nurcan*+

[1] (*)Université Paris 1 - Panthéon - Sorbonne Centre de Recherche en Informatique 90, rue de Tolbiac 75634 Paris cedex 13 France

[2] (+)IAE de Paris - Sorbonne Graduate Business School - Université Paris 1 - Panthéon - Sorbonne 21, rue Broca 75005 Paris France

Telephone : 33 - 1 53 55 27 13 Fax : 33 - 1 53 55 27 01,

Oumaima.Saidani@malix.univ-paris1.fr, nurcan@univ-paris1.fr

Abstract. Security of an information system is becoming an increasingly critical issue. Access control is a crucial technique ensuring security. It should be based on an effective model. Even if some approaches have already been proposed, a comprehensive model, flexible enough to cope with real organisations, is still missing. This paper proposes a new access control model, FORBAC, which deals with the following issues: The first one is the adaptability to various kinds of organization. The second one concerns increasing flexibility and reducing errors and the management cost, this is done by introducing a set a components which allow fine-grained and multi-level permission assignment. The paper introduces a framework for evaluating the proposed approach with respect to other related research through views, facets and criteria.

1 Introduction

The security of an information system or a network consists of its protection against unauthorized access or abusive authorized use. It is assumed by three techniques usually used: authentication [25], access control [23] and audit [10]. In this paper we focus our discussion on access control which consists in managing access rights according to the rules specified in the security policies. In a given organization, it can be formally expressed by instantiating an access control model - often a formal language - in order to represent the security policies in a clear and non ambiguous way. Research on access control started in the 1960s with *Mandatory Access Control (MAC)* [6], [4] and *Discretionary Access Control (DAC)* [13] approaches. *RBAC (Role Based Accecc Control)* [8] appeared with multi-user systems in 1970s and were considered as an alternative to *DAC* and *MAC* [23]. In fact, *MAC* is rigid and imposes strong organizational constraints. *DAC* is vulnerable to information leak. In *RBAC* : instead of permissions being assigned directly to users, they are assigned to roles, and then roles are assigned to users. *RBAC* was promoted by the National Institute of Standards and Technology and was recognized as an American standard in 2004 [9]. *RBAC* models

are "policy-neutral" and can express a wide range of security policies including discretionary and mandatory, as well as user-defined or organizational specific policies [9]. A role-based approach for modelling access control is a natural way to reflect organisational structures and to highlight responsibilities assigned to users. Adopting this approach is useful, particularly if it can meet the flexibility and adaptability requirements of new, complex, evolutionary organizations and widely distributed systems, especially organisational, functional and operational requirements. Nevertheless approaches, dealing with role modelling seem insufficient to meet these requirements.

The purpose of this paper is to improve this kind of approach in order to support flexibility and adaptability requirements. In this paper we illustrate the importance of the flexibility and the adaptability requirements when modelling access control and propose a comprehensive access control model aiming to meet the following requirements related to flexibility:
From the administration point of view:
- R1. Great expressive power allowing to express both simple and complex access control policies
- R2. Reduced cost and risks of errors in administration.
From the usage point of view:
- R3. Adaptability to organizational, functional and operational changes
- R4. Adaptability to various modes of human organizational structures such as functional hierarchies, project teams and knowledge networks.
From the system point of view:
- R5. Suitability to distributed systems and collaborative work.
- R6. Increase safety

For dealing with the requirement R1, we propose modelling access control using a basic model which can be augmented with a number of permission assignment sub-models. For meeting the requirements R2 and R3, we introduce, in the one hand, the concepts of *function* and *cluster of objects*, and in the other hand, we propose to exploit these concepts for *multi-level* and *fine-grained permission assignment*. For dealing with the requirement R4 and R5, we introduce the concept of *organisational unit*. Reducing risks of errors in administration (R2) give confidence to adopt distributed systems and collaborative work (R5), in the one hand, and to increase safety (R6), in the other hand.

We have used these requirements for building a reference framework that is composed of facets or views, each of them is characterised by a set of criteria. This reference framework can be used for evaluating or comparing access control models. In this work, we are going to use it to evaluate our approach with respect to the related work.

The paper is structured as follows: Section 2 discusses the requirements cited above and introduces our approach for modelling access control. In Section 3, we propose a reference framework in order to evaluate our approach with respect to the related work. the conclusion of the paper is given in Section 4.

2 The FORBAC Meta-model

2.1 The Basic Model

In this section we introduce our meta-model for access control. It includes the following concepts: *User, Role, Session, Permission, Object, Organisational-unit, Function, Goal* and *Object cluster*. The first fifth concepts have been adapted from *NIST* model [9]. We are going to discuss them in depth.

An organization can be considered as a set of organisational structures which include relevant users that perform functions in order to achieve particular goals. An organisation can adopt various kinds of organization structures, for instance (i) Functional hierarchies (tree structures), which are based on the concepts of user authorities and responsibilities and also division and specialization of work; (ii) Project teams, that can be defined as a social structure composed of a head and members with various competencies, collaborating to perform an action or a project for a time duration and (iii) knowledge networks which can be defined as communities of practice [21] inter connected, under the same "governorship", sharing common frameworks, methods and tools (for example, experts or mediators that suggest strategies). A user can be assigned to many organisational structures having different kinds. An access control model have to be able to represent all kinds of organizational structures.

For modelling functional hierarchies, we use the concepts of role hierarchies of RBAC. In our approach, we define a role as a set of responsibilities allowing a user to execute functions affected to him (c.f. Appendix.1) *Role hierarchies* has been often mentioned [8, 22]. We identify two different semantics related to hierarchies: specialization/generalization (RSH) and organizational (functional) (ORH).

For modelling, teams and communities of practice, we introduce the concept of *organisational unit* which can be considered as a structure gathering users such as teams, communities of practice, departments independently of roles and role hierarchies(c.f. Appendix.2). Organisational units can be organized, using a tree structure in hierarchical organizations, or be structured as a network in team-based organizations. Thus, we identify two kinds of relations between organisational units:
- $include(ou_2, ou_1)$: means that the ou_2 includes ou_1, e.g. *include(Financial Services, Financial Management Reporting)*
- $collaborate(ou_1, ou_2)$: means that ou_1 collaborate with ou_2. e.g. *collaborate (Audit department, Knowledge and information management service)*

Fig.2 presents some departments of a bank[3] : *Audit*; *Corporate Services* including the sections: *Human Resources, Knowledge and Information Management*, and *Security Services*; and *Financial Services* including three teams : *Accounting and Internal Control, Contracts Management and Procurement*, and *Financial Management Reporting*.

The concept of role does not allow to express many access control rules, because users that are assigned to the same role have the same authorizations.

[3] `http://www.bankofcanada.ca/en/hr/departements.html`

We have integrated a new level between roles and permissions, following the approach of [12]. Instead of assigning permissions to roles, we assign them to functions and functions to roles(c.f. Appendix.3).

In order to highlight our motivation behind the use of the concept of function, let us consider the following situations: S1: a new organisation is set up and it proves to be necessary to distribute the responsibilities of each actor differently; S2 : a responsibility has to evolve. For dealing with S1 and S2, current approaches require checking all permission-to-role assignments and modifying them if necessary. This task is time consuming and includes risk of error. However, competitive environments require quick reactions to changes and do not tolerate inaccuracies. In FORBAC, for dealing with S1, we just have to modify some function-to-role assignments, while users keep their roles, with new assigned responsibilities; And for dealing with S2, we just have to modify some permission-to-function assignments while roles keep their functions, with new assigned permissions. Thus, our approach allows adaptation with organisational, functional and operational changes easily, rapidly with less error (requirements R2 and R3).

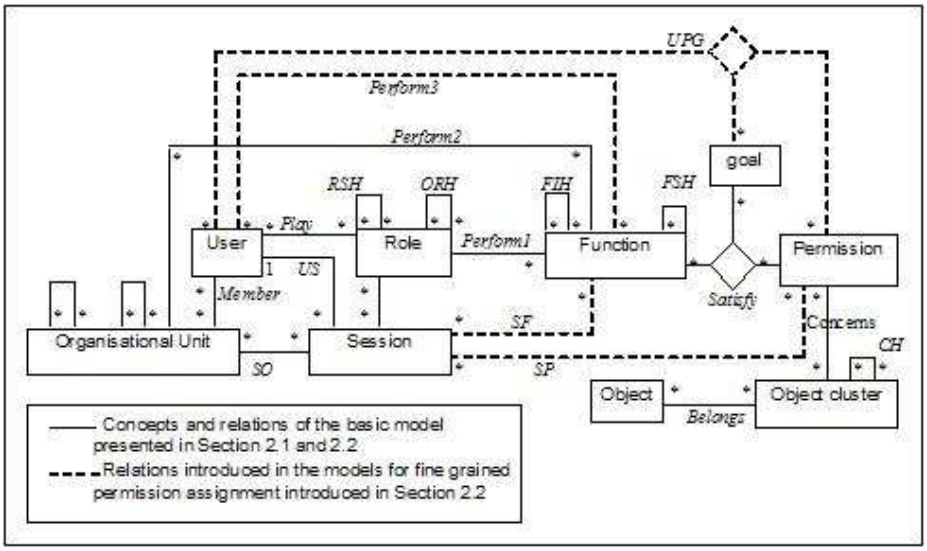

Fig. 1. Illustration of the FORBAC Meta-model using a class diagram

We structure functions in hierarchies. The type of semantics of these hierarchies is either *generalization / specialization* or *inclusion*. Hierarchies of the first type allow specifying most general and abstract functions which are specialized according to particular contexts. For instance, the *financial management reporting* team (c.f. Fig. 2) can be associated with the function "*consulting the financial reporting*" which can be specialized into "*consulting the*

reporting on screen" and "*printing the reporting*". We introduce the predicate $specialized\_function(f_1, f_2)$ which means that f_1 is a *specialized-function* of f_2. Hierarchies of type *inclusion* allow to isolate functions which will be used on several occasions by other functions. For Example, "*managing planning and budgeting*" f_1 includes "*managing planning*" f_2 and "*managing budgeting*" f_3. So, in order to achieve the goal of f_1, we need to achieve f_2 and f_3.

In order to perform functions, users should have permissions for access to resources. We model resources with the concept of *object*. In addition, we introduce the concept of *object cluster* for gathering similar objects by semantic meaning based on different features (user-specified features or attributes) (c.f. Appendix.4). For instance, customers treated by a given adviser. We structure object clusters into hierarchies *(CH)* in order to take advantage of the generalization (up) /specialization (down) properties inherent in the *CH* structure. The mechanism of construction of clusters and cluster hierarchies is out of the scope of this paper. This can be performed using data mining techniques. An object can belong to one or more clusters. This feature allows more coarse grained access control decisions and then help to meet requirements R1 and R2.

We define a permission as an approval of a particular mode of access to particular objects clusters. It confers to the holder of the permission the ability to perform some tasks using these object clusters which can be accessed for various reasons. Permissions allowed for commercial goal are different from those allowed for statistic goals.For this reason we introduce the concept of *goal*.

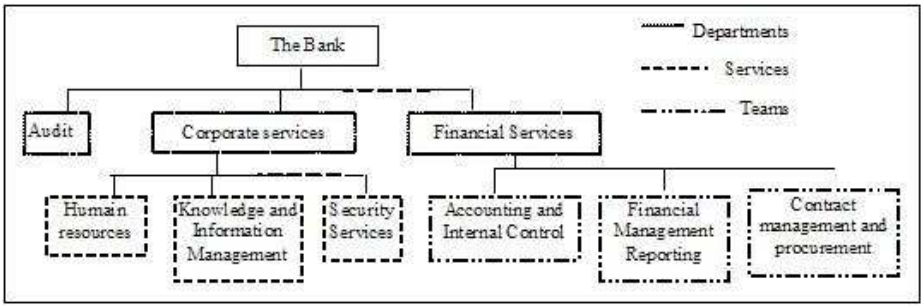

Fig. 2. Example of a hierarchy of organizational units

2.2 Permission Assignment Sub-models

Access control rules depends on particular characteristics of the organisation. Indeed, some rules may be crucial for sophisticated organisations and/or inappropriate for simple ones. A perfect access control model have to be rule-independent and reusable to satisfy needs of a variety of organisations. Although their relevance and benefits, RBAC [9] and its different variants are complicate for some

organisations and/or incomplete for others. Using a composite model including a basic one and a number of assigment sub-models could resolve this problem; Thus, access control policies can be modelled using a selected set of sub-models which includes only the relevant permission assignment possibilities which are specific to a given organisation, rather than using always all the possibilities. A related work in this topic is the NIST RBAC model, which adopts this principle by allowing to integrate only up to four components, nevertheless it is incomplete and imprecise. What is more it does not guide for selecting the most adapted model. That is why we have proposed firstly in Section 2.1 a basic model which can be augmented with a set of assignment sub-models which we are going to discuss in the following.

Basic Model for Permission Assignment. In this paper we focus only on positive permissions, i.e. the right to execute an operation or task. Prohibition, obligation and administrative rights, i.e. the right to manage users, permissions, roles and so on, are not discussed. Permission-to-function assignment relation is formulated using the predicate *Assigned_Permissions* (c.f. Appendix.5).

A user can open many sessions simultaneously. The mapping of a user u) onto a set of of sessions is formulated as follows: $User\_Sessions(u : USERS) \rightarrow 2^{SESSIONS}$. He has to select a subset of roles and organisational units assigned to him. According to this selection, a particular set of permissions is granted. The permissions available to the user are the union of permissions from all roles and organisational units in this session. Each session is a mapping of one user to a subset of roles and organisational units. (c.f. Appendix.6).

We define inheritance between roles as follows: For the specialization / generalization hierarchies, we introduce the predicate $Specialized\_role(r_1, r_2)$ which means that r_1 specializes r_2, it is a partial order on $ROLES$, written as $\geq$, where $r_1 \geq r_2$ only if all permissions of r_2 are also permissions of r_1, and all users of r_1 are also users of r_2. (c.f. Appendix.7).

For the organizational hierarchies of roles. We introduce the predicate $Junior\_role$ (r_1, r_2) which means that r_1 is hierarchically inferior to r_2. It is a partial order on $ROLES$, written as $\geq$, where $r_1 \geq r_2$ does not signifies always that all permissions of r_2 are also permissions of r_1, nor all users of r_1 are also users of r_2. For example (c.f. Fig.2): *the director of the corporate services does not inherit all permissions of an employee in security services.* (c.f. Appendix.8).

The inheritance through the inclusion hierarchies of organisational units is represented in the following rule: $Include \subseteq ORG\_UNITS \times ORG\_UNITS$ is a partial order on ORG_UNITS, written as $\geq$, where $ou_1 \geq ou_2$ does not signifies always that all permissions of ou_2 are also permissions of ou_1, nor all users of ou_1 are also users of ou_2. (c.f. Appendix.9).

The relation of inheritance through hierarchies of functions is a partial order on $FUNCTIONS$ written as $\geq$, where $f_1 \geq f_2$ only if all permissions of f_2 are also permissions of f_1, and all users of f_1 are also users of f_2. The inheritance of the permissions associated with this hierarchy is defined using the predicate Specialiozed_function (c.f. Appendix.10).

We introduce the predicate $Included\_function(f_1, f_2)$ which means that f_1 is used by f_2. This relation is also a partial order on $FUNCTIONS$ written as $\geq$, where $f_2 \geq f_1$ only if all permissions of f_1 are also permissions of f_2, and all users of f_2 are also users of f_1. The inheritance of the permissions associated with this relation is formulated in Appendix.11.

We introduce the predicate $Junior\_Cluster(c_1, c_2)$ which means that c_1 is junior to c_2. $Junior\_Cluster$ a partial order on $CLUSTERS$ written as $\geq$.where $c_2 \geq c_1$ only if all permissions on c_2 are also permissions on c_1, and all users allowed to access to c2 are also allowed to access to c_1. The inheritance of permissions is formulated in Appendix.12.

Models for Fine Grained Permission Assignment

Function-to-User Assignment. A user can be directly assigned to particular functions and select a subset of them in a given session. According to this selection, a particular set of permissions is granted. So, each session is a mapping of one user to a subset functions which can be activated and deactivated at the user's discretion. (c.f. Appendix.13).

Permission - to - user assignment. In a session, a user can select a subset of permissions which are directly assigned to him.According to this selection, a particular set of permissions is granted. In a session, permissions, can be activated and deactivated at the user's discretion. Each session is a mapping to a set of permissions. (c.f. Appendix.14).

Permission_to_User and Permission_to_User assignements allow fine grained assignements offering greater flexibility in handling permissions such as handling exeptions.

Composed Permission Assignment Model. According to the access control roles, we can use an integrated model to model muti-level and fine-grained permission assignments. The integrated model can be composed of the basic model for permission assignment augmented with a number of assignment models presented above. In a session, a user can select a subset of roles, organisational units, functions and/or permissions. According to this selection, a particular set of permissions is granted. The permissions available to him are the union of permissions from all roles, organisational units, functions and/or permissions activated in that session and granted directly or indirectly to him. Each session is a mapping of one user to a subset of roles, functions, org-units and/or permissions. Roles, org-units, functions and permissions, in a session, can be activated and deactivated at the user's discretion.

3 Evaluation of FORBAC with Respect to Related Work Using a Reference Framework

In order to characterize our model with respect to the related work, we developed a reference framework relying on [16]. This framework, initially proposed for sys-

tem engineering [14], proved its effectiveness in the comprehension improvement of many disciplines of engineering, such as IS [14] and requirement engineering [15]. The approach "by facets" was first proposed in [21]. It was also used in the process engineering domain [24] to compare organizational change management approaches [3, 19]. A facet includes a set of attributes allowing to specify the characteristics of the studied approaches and to build then a comparative state of the art. We identified three views (Administration, Usage and System). Each view is characterized by a set of facets facilitating the understanding and the classification of the studied access control models. The facets identified are: *environment* and *safety* (System view); *type of administration, expressiveness, complexity* and *Cost* (Administration view); *adaptability to changes*, and *adaptability to organisational structures* (Usage view). Some facets include a set of attributes allowing to specify the characteristics of the studied models and to build then a comparative state of the art. For example, the facet complexity in the administration view has an attribute named *level*. Figure 3 represents our framework, it's facets, the attributes of some facets, and the set of possible values of attributes. Figure 3 characterize also our model with respect to the framework. The underlined values, specify the characteristics of our model.

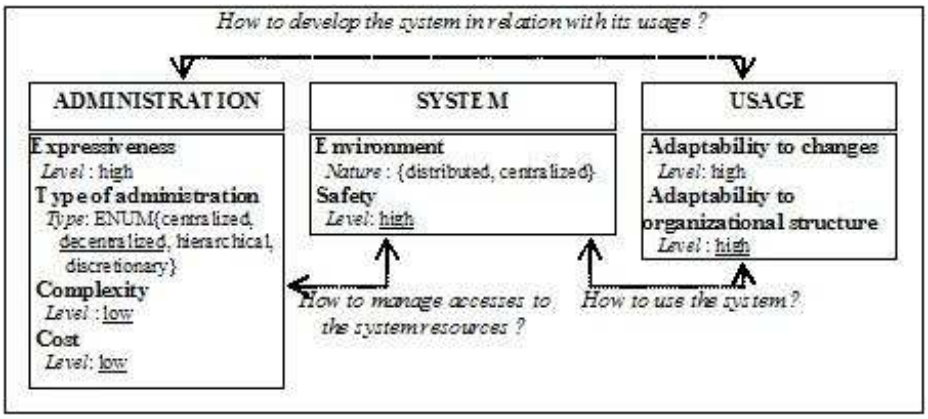

Fig. 3. The Reference Framework used to characterize the proposed model

3.1 System View

Environment. Our model satisfies widely distributed system needs. It is adaptable to various organization modes, thanks to the concept of organizational unit. Classifying resources into clusters helps to easily decentralize them.
Safety. Introducing the concepts of function and object cluster allows to precisely specify permissions and then to reduce error risks when managing access control, sequentially. In consequence, safety is better assured.

3.2 Usage View

Adaptability to organizational structure and to changes. Hierarchies are suitable to organize roles and to reflect skills and duties into organizations. Nevertheless, this concept is ambiguous in *RBAC* and majority of its variants [5, 8, 11, 17, 9, 22]. The permissions inheritance is sometimes incorrect. For example, based on Figure 2, a user, playing the role of *director of the corporate services*, r_1, is hierarchically superior to a user playing the role of *employee in security services*, r_2, however, r_1 should not inherit all r_2's permissions. In our approach, we use the concept of roles hierarchies to structure roles, but we distinguish two distinct semantics for roles hierarchies : the specialization/generalization hierarchies and the functional or organizational hierarchies. Current models do not distinguish between these two hierarchy semantics. What is more, they do not support issues related to adaptability to various organisations and to changes.

3.3 Administration View

Expressiveness As discussed in Section 2.2, our approach allow managing permissions at different levels of granularity. This feature increases flexibility and allows to express a broad range of security requirements from simple to complex. The concept of goal and the user-to-function and user-to-permission assignment relations give to the model the ability to create more flexible instances (the latter will describe the access control policies of real organizations in a changing environments). *RBAC* is somewhat non flexible for granting specific rights, since it only permits granting rights by defining an appropriate role and assigning users the right to use it. It is appropriate only to organizations whose users are assigned to roles with well defined access rules [9]. *TMAC* [11] introduces the concept of *team*. *C-TMAC* [11] provides explicitly activation permission rules according to the context. *ORBAC* [18] focuses on the concept of organization. *TRBAC* [5] introduces the concepts of role periodic enabling and *temporal dependencies* between roles. *GTRBAC* [17] allows specifying temporal constraints on role enabling and temporal restrictions on the user-to-role and role-to-permission assignments. *TBAC* [26] extends *RBAC* with the concepts of *task*, *authorization step* and *authorization step life cycle* concepts allowing subjects to dynamically obtain permissions while performing tasks. Most of the existent models are restricted to permissions. *ORBAC* [18] distinguishes three types of privileges: permissions, prohibitions and obligations. This mixed policy can cause problems related to conflict management and redundant rules. Related works cited are less expressive than our approach since they do not allow adaptability with changes and various types of organisations.

Complexity Permissions can be modified either by explicit authorizations of a user to a role, by changing the set of functions of a role or by changing the set of permissions of a function. Our model is less *complex* to administrate than the others thanks to the operations provided for role handling; the roles, organisational units and functions hierarchies allow us to partially automate user-to-role,

role-to-function and function-to-permission assignments. *FORBAC* has multiple advantages compared to *RBAC* (and to quote the other models). Indeed, permission-to-function assignments and function-to-role assignments reduce considerably the total number of permission-to-role assignments as follows : For each function, let R be the number of roles exerting a function an P the number of permissions required for a function: $(R+P) < (R \bullet P), R, P > 2 \Rightarrow (R+P) < (R \bullet P)$. For all job positions, $\sum_i^{n_{jp}}(R_i + P_i) < \sum_i^{n_{jp}}(R_i \bullet P_i)$.

The concepts of function and object cluster allow to reduce management cost, in fact, fine-grained and multi-level permission assignment allow to suit actual organisation requirements. As opposed to assignment at the role level in the *RBAC* model.

Type of administration There are several kinds of administration: it is discretionary in *DAC*, centralized in *MAC* and decentralized in *RBAC* and *ORBAC*. *ARBAC* and *AdOBAC* [7] are designed to manage respectively *RBAC* and *ORBAC*. As far as we know, there are no administrative models proper to *TMAC*, *C-TMAC*, *TBAC*, *TRBAC* and *GTRBAC*. However, being based on *RBAC*, they can be managed by *ARBAC*.

4 Conclusion and future work

We have proposed a new approach for modelling access control, named *FORBAC*, aiming to overcome some weaknesses of the current ones and to meet flexibility requirements in evolutionary environments. We have formally defined the approach's basic concepts and the way permissions can be inherited. We have evaluated our approach with respect to the studied models; that's why we have proposed a reference framework that we have used as basis for evaluating access control models. We think that our approach is not only expressive, but it is also sufficiently flexible and adaptable to suit to evolutionary requirements of the organisations. Indeed, it allows a great expressive power thanks to the use of a basic model which can be augmented with additional permission assignment sub-models; Thus, *FORBAC* allows several levels for assigning permissions to users. Furthermore, *FORBAC* use the concept of *organisational unit* in order to suit to various types of organisation structure and to disributed systems and collaborative work. In addition, we have introduced the concepts of *function* and *object cluster*, and their use for multi-level and fine grained permision assignment help to adapt easilly to organisationa, functional and operational changes with reducing the risk of errors in administration which increase safety.

Several issues have not been discussed here but they will be presented in our future work:
- The translation of users from one organisational unit to another, logically they have not to keep all rights related to the previous tasks.
- The application to a workflow, that means the change of permissions based on the state of an object.
- Other types of relations between functions need to be discussed, like "*decomposition*".

Modelling this type of decomposition is an important issue in several applications, particularly for business process applications. For example, a function f1 can be decomposed into f_2 followed by f_3. If a given role r has the permission to perform f_1, then it should also have permissions to perform f_2 and f_3. However, he should have the permission to perform f_3 only after having performed f_2.
- It seems necessary to develop an administrative model for *FORBAC*. Such a model has been proposed with *ARBAC* [20] for administrating *RBAC*.
- We also try to define a mechanism allowing to guide the administrator to use the appropriate model in order to meet better with the security organization requirements. Such a model have to include just the adequate access control model components and have to be able to manage the security policy changes following organizational, operational and functional changes. This will lead to improve safety and reduce administration cost.

References

[1] G. J. Ahn et R. Sandhu. Role-based Authorization Constraints Specification. A*CM Trans. Inf. and Sys. Sec.*, 3(4), 2000.

[2] E. Barka et R. Sandhu. A role-based delegation model and some extensions. *NISSC*, 2000. *ACM Trans. Inf. and Sys. Sec.*, 4(3):191-233, 2001.

[3] Barrios, J.: Une méthode pour la définition de l'impact organisationnel du changement, Thèse de Doctorat de l'Université Paris1 (2001)

[4] D. E. Bell et L. J. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Technical Report ESD-TR-73-306, The MITRE Corporation, 1976.

[5] E. Bertino, P.A. Bonatti et E. Ferrari. TRBAC:A Temporal Role-Based Access Control Model .

[6] K. J. Biba. Integrity for secure computer systems. Technical report MTR-3153, The MITRE Corporation. *ACM Trans. Inf. and Sys. Sec.*, 4(3):191-233, 2001.

[7] F. Cuppens et A. Miège. Administration model for Or-bac. International Federated Conferences (OTM'03), Workshop on Metadata for Security. Italy, Nov. 3(7): 754-768, 2003.

[8] D. Ferraiolo et R. Kuhn. Role-Based Access Control. Proceedings of 15th NIST-NCSC National Computer Security Conference, 554-563, Baltimore, MD, 1992.

[9] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn et R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. ACM Trans. Inf. and Sys. Sec., 4(3):222-274, 2001.

[10] Frederick, G., Daniel, M., Sandra, S., Carol, G.: Information Technology Control and Audit. Auerbach publications (2004)

[11] C. K. Georgiadis, I. Mavridis, G. Pangalos et R. K. Thomas. Flexible Team-based Access Control Using Contexts. ACM RBAC Workshop, Chantilly, VA USA 2001.

[12] Goncalves, G., Hémery, F.: Des cas d'utilisation en UML la gestion de rôles dans un système d'information. Actes du Congrès INFORSID, France (2000).

[13] Harisson, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in Operating Systems. Communication of the ACM, 19:8 (1976) 461-471

[14] M. Jarke, J. Mylopoulos, J.W. Smith et Y. Vassilio. DAIDA - An environment for evolving information systems. ACM Trans. on Inf. Sys., 10(1). 1992.

[15] Jarke, M., Pohl. K.: Requirement engineering : an integrated view of representation, process and domain. Proc. of the 4th European Soft. Conf., Springer, 1993.
[16] M. Jarke, C. Rolland, A. Sutcliffe, R. Dömges, The NATURE Requirements Engineering, Shaker Verlag, Aachen, ISBN 3-8265-6174-0. 1999.
[17] J. B.D. Joshi, E. Bertino, U. Latif, et A. Ghafoor. A Generalized Temporal Role-Based Access Control Model. IEEE Transactions on Knowledge and Data Engineering, 17(1), 2005.
[18] A. E. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel et G. Trouessin. Organisation Based Access Control. POLICY'2003. Italie, 2003.
[19] Nurcan, S., Barrios, J., Rolland C.: Une méthode pour la définition de l'impact organisationnel du changement. ISI, N spécial, INFORSID (2002)
[20] S. Oh et R. Sandhu. A Model for Role administration using Organization Structure. In Proc. of the 7th ACM SACMAT, California, 155-162, 2002.
[21] Prieto-Diaz, R., Freeman, F.: Classifying software reusability. IEEE Software (1987)
[22] R. Sandhu. Future Directions in Role-Based Access Control Models. MMM-ACNS 2001.
[23] Sandhu, R., Coyne, E., Feinstein, H., Youman, C. E.: Role Based Access Control Models. IEEE Computer, 29:2 (1996) 38-47
[24] Si-Said Cherfi, S.: Proposition pour la modélisation et le guidage des processus d'analyse des systèmes d'information. Thèse de Doctorat Université Paris 1 (1999)
[25] Smith, R. E.: Authentication From Passwords to Public Keys. Addison Wesley, 2002.
[26] R. Thomas et R. Sandhu. Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management. 11th IFIP Working Conference on Database Security, Lake Tahoe, USA, 1997.

A Appendix

We are going to express basic concepts and permissions-assignment of FORBAC in a formal functional specification based on first-order logic

1. $Play \subseteq USERS \times ROLES$, $play(u, r), u \in USERS, r \in ROLES$, means that u can play r.
   $Assigned\_Users1(r : ROLES) \rightarrow 2^{USER}$ (adapted from the RBAC model, maps a role r onto a set of users)
   $Assigned\_Users1(r) = \{u \in USER/(u, r) \in Play\}$
2. $Member \subseteq USERS \times ORG\_UNITS, member(u, ou), u \in USERS$,
   $ou \in ORG\_UNITS$, means that the user u is a member of the org-unit ou.
   $Assigned\_Users2(ou : ORG\_UNITS) \rightarrow 2^{USER}$ maps an org-unit ou onto a set of users.
   $Assigned\_Users2(ou) = \{u \in USER/(u, ou) \in Member\}$
3. $Satisfies \subseteq PERMISSIONS \times FUNCTIONS \times GOALS, Satisfies(p, f, g)$, $p \in PERMISSIONS, f \in FUNCTIONS, g \in GOALS$ means that f is authorized to p in order to achieve g.
   $Assigned\_Function(r : ROLE) \rightarrow 2^{FUNCTION}$, is a mapping of a role r onto a set of functions.
   $Assigned\_Function(r) = \{f \in FUNCTION/(f, r) \in Perform1\}$

4. $Belongs \subseteq OBJECTS \times CLUSTERS$, an object-to-cluster assignment.
   $Objects\_of\_cluster(c : CLUSTER) \rightarrow 2^{OBJECT}$ the mapping of cluster c onto a set of objects
   $Object\_of\_cluster(c) = \{o \in OBJECT / (o, c) \in OC\}$.
5. $Assigned\_Permissions(f : FUNCTION, g \in GOAL) \rightarrow 2^{PERMISSION}$
   $Assigned\_Permissions(f, g) = \{p \in PERMISSION, g \in GOAL) / (p, g, f) \in Satisfies\}$
6. $Sessions\_Role(s : SESSIONS) \rightarrow 2^{ROLES}$
   $Session\_Roles(S_i) \subseteq \{r \in ROLES / (User_Session(S_i, ou) \in Play\}$
   $Sessions\_Org\_units(s : SESSIONS) \rightarrow 2^{ORG\_UNITS}$
   $Session\_Roles\_units(S_i) \subseteq \{ou \in Org_units / (User\_Session(S_i), ou) \in Member\}$
7. $\forall r_1, r_2 \in ROLES, p \in PERMISSIONS, u \in USERS, g \in GOALS$
   $Specialized\_role(r_1, r_2) \wedge Authorized(r_2, p, g) \rightarrow Authorized(r_1, p, g)$
   $Specialized\_role(r_1, r_2) \wedge Play(r_1, r_2) \rightarrow Play(r_1, r_2)$
8. $\exists r_1, r_2 \in ROLES, u \in USERS, p \in PERMISSIONS, g \in GOALS$
   $Junior\_role(r_1, r_2) \wedge Authorized(r_2, p, g) \wedge \neg Authorized(r_1, p, g)$
   $Junior\_role(r_1, r_2) \wedge Play(u, r_1) \wedge \neg Play(u, r_2)$
9. $\exists ou_1, ou_2 \in UNIT\_ORG, u \in USERS, p \in PERMISSIONS, g \in GOALS$
   $Include(r_1, r_2) \wedge Authorized(ou_2, p, g) \wedge \neg Authorized(ou_1, p, g)$
   $Include(ou_1, ou_2) \wedge Member(u, ou_1) \wedge \neg Member(u, ou_2)$
10. $\forall f_1, f_2 \in FUNCTIONS, p \in PERMISSIONS, g \in GOALS$
    $Specialized\_function(f_1, f_2) \wedge Authorized(f_2, p, g) \rightarrow Authorized(f_1, p, g)$
11. $\forall f_1, f_2 \in FUNCTIONS, p \in PERMISSIONS, g \in GOALS$
    $Included\_function(f_1, f_2) \wedge Satisfies(f_2, p, g) \rightarrow Satisfies(f_1, p, g)$
12. $\forall c_1, c_2 \in CLUSTERS, p \in PERMISSIONS$
    $Authorized(c_2, p) \wedge Junior\_Cluster(c_1, c_2) \rightarrow Authorized(c_1, p)$
13. $Sessions\_Functions(s : SESSIONS) \rightarrow 2^{FUNCTIONS}$
    $Session\_Functions(S_i) \subseteq \{f \in FUNCTION / (User\_Session(S_i), f) \in Perform\}$
14. $Sessions\_Permissions(s : SESSIONS) \rightarrow 2^{PERMISSIONS}$
    $Session\_Permissions(S_i) \subseteq \{p \in PERMISSIONS / (User\_Session(S_i), p) \in UPG\}$