International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 10, No.1 (Nov-2021)

http://dx.doi.org/10.12785/ijcds/1001103

Towards An Extension Of RBAC Model

Abdelkrim BOUADJEMI[1] and Mustapha Kamel ABDI[1]

[1] Université Oran1 Oran, Algérie

Received 12 July.2020, Revised 22March. 2021, Accepted 3April. 2021, Published 25 Nov. 2021

Abstract: SystemsRole-based access control (RBAC) has been widely used in information systems including so-called critical systems. Access control models describe the frameworks that dictate permissions. The RBAC model is generally static, ie., access control decisions are: grant or deny. This model is effective in normal situations. In other situations, such as exceptions or emergencies, flexible access control is required. In order to increase the flexibility of access control, the concept of obligation has been proposed. Obligations are requirements to be fulfilled in order to execute permission decisions.
The purpose of this article is to produce a flexible model which uses obligations to manage exception situations. Our model increases the flexibility of the RBAC model. It allows to assign permissions dynamically.For illustration, Anderson's clinical information system is used. Finally, Alloy is used to analyze the validity of the proposed model.

Keywords:RBAC, Exception, Constraints, Obligations, Permissions, Formal Specification, Alloy, Consistency.

1. Introduction

With the development of information systems (IS), the functional specification without taking into account the security requirements exposes these IS to risks. Access control is an essential element of IS protection. Most organizations define roles for different organizational tasks. RBAC is the most appropriate for these organizations and the most used for IS in terms of access control. In 1996, Sandhu et al. in [26] proposed role-based access control and introduced four different models: the basic model, the hierarchical model, the constraint model and the consolidated model. The National Institute of Standards and Technology (NIST) proposed NIST RBAC to address the lack of a model that creates uncertainty and confusion about the usefulness and meaning of RBAC [8]. The modification of NIST RBAC by National Standard for Information technology (ANSI) was made by defining a new standard for RBAC[2].

In reality,sometimes the access control policy is insufficient because users or the system can also perform actions or provide information that is not provided for in the policy.However, access control models lack flexibility. In the literature, the mechanisms of delegation and Break The Glass (BTG) have been proposed to increase flexibility [6],[23],[30]. In the RBAC context, role permissions allow access to tasks. The main contribution of this article is the proposal of an Exception RBAC (Exc-RBAC) model to manage the system in situations of exceptions with respect to existing constraints.

This work proposes to improve the flexibility of access control to allow IS to adapt to different situations, i.e., a role that does not have a given permission may request it in an exception situation but under certain conditions. The permission requested by the user must not cause a conflict between the roles activated in a session. The proposed model uses the obligation in order to authorize a reject decision. Obligations are requirements to be met, which can be extended to conventional systems in order to specify additional information. Among the contributions of this article, the authors introduced the notion of dynamic separation of duties (DSD) as a precondition; if the precondition is not true, then the obligation is not fulfilled and the requested permission cannot be accepted.

In Exc-RBAC, user assignments to roles and role permissions verified by an administrative model. Administrative role-based access control models such as ARBAC97 are used to manage large RBAC systems[27]. ARBAC97 controls role assignment to the user, permission to a role, and addition or deletion of roles in the role hierarchy. To handle exception situations, we have extended ARBAC97 to control the assignment of

E-mail address: abdelkrimbouadjemi@yahoo.fr, abdimk@yahoo.fr

roles and permissions. For illustrative purposes, we used Anderson's clinical information system as an example [1]. Then, the specifications of model and example were implemented via Alloy [13]. Alloy is a first-order logic-based modeling tool and is a high-level language. It is used for the specification and verification of access control policies. Alloy Analyzer is used by Alloy whose purpose is to create micro-models for automatic model validation. In the present work, the proposed model and the example are verified using Alloy analyzer.

In the rest of the paper, section 2 reviews the literature on the issue. Section 3 present RBAC model and the basic RBAC with augmentation of obligations. In section 4, we begin by presenting model, then show how integrate the administrative model in proposal. In section 5, the formal specification of Exc-RBAC is implemented in the Alloy Analyzer tool and its consistency is checked. In section 6, the clinical information system example illustrates this work. Finally, the last section finishes the paper and furthermore presents some perspectives

2. RELATED WORK

The consideration of access control policies is a major concern. Among the types of access control, the most used is RBAC. In [7], the authors formally specify a model called Smatch (Secure MAnagement of swiTCH). In the model,dynamic session is created by the user. The dynamic session allows sharing, reusability and permutability. User can run task, suspend and restart session by running this task in another context. In [4], the authors propose an extension of the RBAC model named Temporal-RBAC (TRBAC). The functionality of TRBAC supports the time constraint for periodic activation and deactivation of roles and individual exceptions. Cotrini et al. in [5] present FORBAC (First Order Role Based Access Control), an extension of first-order role-based access control. The authors propose to transform requests to the issue of Satisfiability Modulo Theories (SMT).

Lu et al. in [17] are interested in the problem of the User Authorization Request (UAQ) related to processing user access requests in RBAC. The authors study the problems related to the reassignment of roles and permissions whose purpose is to determine whether a given assignment of roles and permissions satisfies all the reassignment objectives and does not violate any prior constraint or constraint of authorization capacity. Subramanian et al. in[35] represents the RBAC access control matrix in the form of three-way concepts: permit, deny and non-commitment, to allow verification of roles hierarchy and constraints of RBAC.The authors in [3] propose a process which corrects an RBAC state when the system detects an exception. Exceptions are permissions that few users don't have that are indispensable for their work and ought to be allowed at the earliest opportunity. The authors concern was to adjust two objectives : simplifying the present state of the RBAC and decreasing the expense of transition.

Liu et al.in [16] proposes a mechanism which makes changes in user permission called transformation. The authors integrated the transformation into the BTG-RBAC model to create the Ts-RBAC model. So, depending on the situation, users are assigned different permissions. In [19], the authors proposed the emergency RBAC (E-RBAC) whose aim is to reinforce flexibility, this approach uses the BTG policy and the separation of tasks (SOD) is included whose aim is to control the 'user access in emergency situations. The authors also used the administrative model in a large E-RBAC system to manage access control. Alloy was used to implement the model specifications and the medical scenario. In this article, we propose an extension of the RBAC model for exception situations. The approach proposed is based on the reassignment of permissions to roles.

In RBAC, a role has one or more specific permissions.However, sometimes users need to access unauthorized resources under normal situation. In order to increase the flexibility of access control and solve this problem, BTG and delegation have been proposed. For flexible access control management, the authors in [32] presented a review of BTG model and Delegation. Delegation is a mechanism of assigning access rights to a user. The authors in [6],[23],[30]studied transfer delegations for RBAC models. They include grant delegation in their model for incompleteness, and they presented many tools which allow delegations in their model. As result, they have shown that using relationships for authorization of delegations is less efficient than using administrative scope.

BTG policies are flexible and its principle is to allow users to interrupt or bypass access controls in a controlled manner. The principal goal of Ferreira et al. in [9]is to extend NIST/ANSI RBAC model by BTGRBAC model so that it can be adopted when unforeseen or emergency situations arise[18]. There are a few BTG and delegation models involving constraints such as separation of duty (SOD) and binding of duty (BOD), which are added to access control decisions [31],[34],[37]. The author in [14] suggests to integrate tasks into RBAC systems, these tasks must be performed by users, this approach is very similar to the notion of obligation.

In[39], the authorsintroduced RBAC augmented with obligations, the PA relationship in the NIST RBAC model allows privileges to be assigned to roles, these privileges are permissions associated with roles.In order to take into account obligations, roles can be associated with obligations as well as permissions, such that each obligation or a set of obligations is associated with a permission which is allocated to a role.

In [20], the authors proposed a model of obligations for P-RBAC (core Privacy-aware Role Based Access Control). The obligation model supports: pre-obligations, post-obligations, conditional obligations, and repeating obligations. In [24], the authors are interested in obligations that require permission and when they are executed, they can change the status of permission. The approach proposed in this article is based on obligations, which allow specifying requirements in order to grant or refuse a requested permission to a role in a session. This work proposes to integrate the ARBAC97 administrative model to control the allocation of permissions in exception situations.

Self-management is one of the benefits of RBAC, but with the existence of a large number of users, roles and permissions, it will be too complicated for a security administrator to manage these components.[29]. Administrative models have been proposed as a solution to this problem. In [27], Sandhu et al. presented an administrative model called ARBAC97 for RBAC. It has three components: URA97 concerns user-to-role assignment; PRA97 with permission-to-role assignment; and RRA97 deals with role-to-role assignment.In the literature, ARBAC99 [28], ARBAC02 [21], [22], ARBAC07 [38] and AMTRAC [33] are part of the family of administrative models.

In the universe of current access control systems, definition of an access control policy is to specify knowledge in formal language [25]. The authors in [16]proposed a classification of situations into three categories to ameliorate the flexibility of the systems:

- Normal situation: possibility of defining the access control policy as well as the user access information are known.
- Emergency situation: possibility of defining the access control policy as well as the obligation to determine the necessary accesses.
- Exception situation: impossibility of defining the access control policy and the necessary access is unknown.

3. RBAC MODEL AND OBLIGATIONS

Sandhu et al. in [26] have developed a model that uses roles as essential constructors in access control models. The authors designed RBAC for four models: the base model, the hierarchical model, the constraint model, and the consolidated model. The first model called Flat RBAC (base model), is the basis from which the hierarchical model and the constraint model have been developed, it is the basic concept that allows systems to support RBAC. The second, called Hierarchical RBAC adds the concept of role hierarchy, in the case of inheritance of permissions between roles. The third, called Constrained RBAC, allows to add constraints. The last one called Consolidated model, is the combination of hierarchical models and constraints. The connection between these four models is shown in figure 1

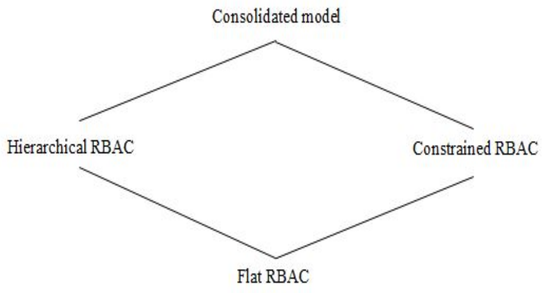


Figure 1. Relationship between RBAC models

The Core RBAC [2] consists of five basic elements, which are the Users, Roles, Operation, Object, and Sessions, and five relations, which are the User Assignment (UA), the Permission Assignment (PA), the Users Session (US), the Session Roles (SR) and the Permissions. The model is illustrated in Figure 2.

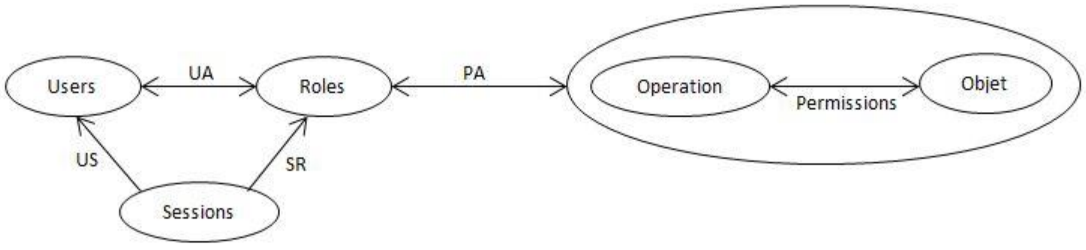


Figure 2. The RBAC model

In order to increase flexibility, the notion of obligation has been incorporated into the RBAC model. The idea is that obligations can be affected to roles with permissions, as a group of roles which is associated with a role for each permission. Permission assigned to different roles may be associated with the same obligation or different obligations [39].

Figure 3 illustrates the extended RBAC model (with obligation), this model includes a new basic element, OBLGS, which is the set of valid obligations. These obligations represent tasks that can be executed by the system and associated with permissions.

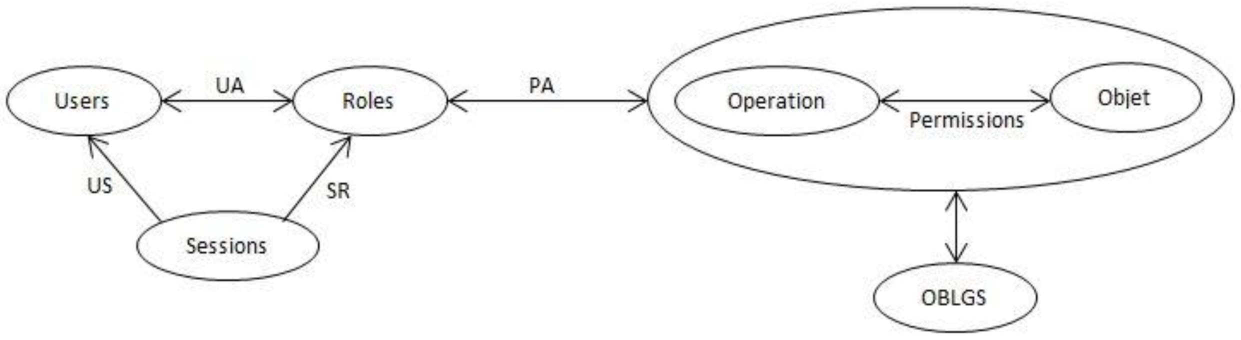


Figure 3. The RBAC model with obligations

4. EXCEPTION RBAC «EXC-RBAC»

A. Definition of model

Figure 4 shows the model Exc-RBAC

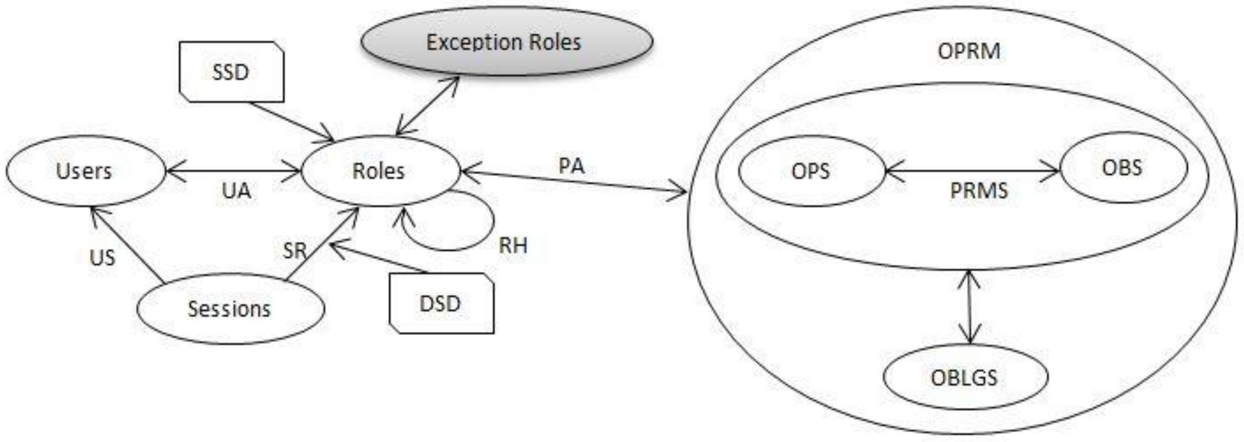


Figure 4. Exception RBAC

Our model incorporates RBAC96 and obligations which are defined by six equations.

$$PRMS \subseteq OPS \times OBS \quad (1)$$

$$UA \subseteq Users \times Roles \quad (2)$$

$$PA \subseteq Users \times OPRMS \quad (3)$$

$$US(s:session) \rightarrow Users \quad (4)$$

$$SR(s:session) \rightarrow 2^{Roles} \quad (5)$$

$$OPRMS \subseteq PRMS \times 2^{OBLGS} \quad (6)$$

The permission function produces a response containing the boolean type permission decisions and associated obligations as expressed as:

$$CheckAccess(s,op,obj) = \exists\, r \in Roles,\ r \in SR(US(s)) \wedge ((op,ob),r) \in PA) \quad (7)$$

In an exception situation, the information on the required access is unknown, with the impossibility of determining the policy. The operation of the Exc-RBAC model is illustrated in algorithm 1. When the user goes into an exception situation, he can request permission from a hierarchically superior role. First, the user defines the state with exception and establishes a session, then he is informed of the obligations related to the administrative role. For permission to be granted to the user, this latter must fulfill the obligations and the administrative role must verify that the role of the user and the role of the permission must not be dynamically contradictory.

```
user set system statut to Exception
user establish session
obligation(s) must be satisfy
user U requests permission P
Begin
  If (user role U ∩ role of permission P) ∉ DSD then
    Begin
      If obligation(s) is satisfy then
        assign permission to role
    End
End
Delete permission of role
End session
```

Algorithme 1. Exc-RBAC algorithm

B. *Administrative model for Exc-RBAC*

The administrative model was proposed as a solution to manage a system that contains large number of users, roles and permissions. Among the administrative models that manage large RBAC models, there is ARBAC97. The ARBAC97 model was proposed to control the assignment of user roles, permissions to roles as well as the addition or deletion of a role in the role hierarchy.It consists of three models: URA97, PRA97 and RRA97 which administers user-to-role, authorization-to-role, and role-to-role assignment respectively;as well as the role range, administrative role and normal role [27].In order to control the assignment of roles and permissions in exceptional situations, we have extended ARBAC97.In this article, we extend ARBAC97 in exception situation in order to support Exc-RBAC.

The administrative role is informed of exception situations by the user who requests permission to resolve them.In exceptional situations only permission to role assignment occursby modifying PRA97 but URA97 and RRA97 remain unchanged. PRA97 allows assigning permissions to roles and revoking permissions. The permissions to roles assignment are controlled by can_assignp(ar,pc,r), where aris an administrative role, r is the user roles that request permission, and pc is a prerequisite condition. The revocation of permission is controlled by can_revokep(ar,r), where aris an administrative role and r is a role range. In the prerequisite condition, DSD constraints are verified. In an exception situation, the defined functions are used along with assign-roletop(p). This function returns the roles of requested permission p formulated as follows:

$$\text{assigned\_roletop}(\ p : \text{PRMS}\ ) \rightarrow 2^{\text{ROLES}} \tag{8}$$

In an exceptional situation, when a user U requests a permission P, the administrative role ar uses the relation can_assignp (ar, pc, r) in which pc is a precondition expressed as follows:

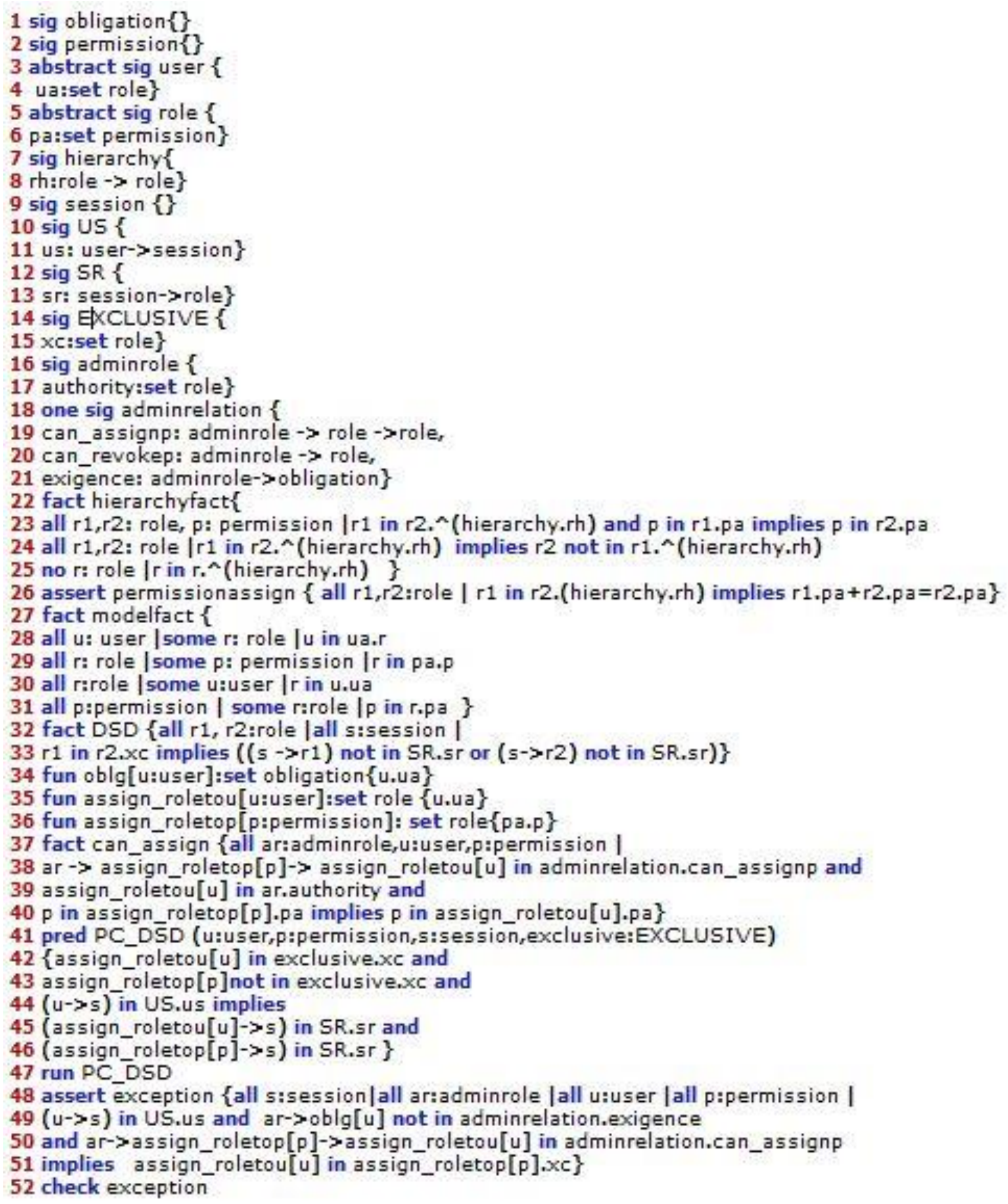
$$\text{assign\_roletou(u)} \cap \text{assign\_roletop(p))} \notin \text{DSD} \tag{9}$$

In the prerequisites conditions, the intersection of the roles returned by the assign_roletou () and assign_roletop () functions must not be in DSD, i.e.,the two roles should be inactivated in the same time in a session for the same user. If the precondition is met, the user's role is granted permission.At the end of the exception, the relation can_revokep (ar, r) is used by the administrative role to remove the P permission from the role r.

5. IMPLEMENTATION AND EVALUATION OF FORMAL SPECIFICATION EXC-RBAC

Alloy is developed by the Massachusetts Institute of Technology (MIT) Group Software Design. Alloy is a first-order logic-based modeling tool and is a high-level language for expressing constraints. Alloy uses the Unified Modeling Language (UML) package, the subset selected by Alloy in UML is consistent [36]. Alloy Analyzer is used by Alloy whose purpose is to create micro-models for automatic model validation.

In this section, Figure 5 illustrates the formal specifications of our Exc-RBAC model which are defined in Alloy.



```
1 sig obligation{}
2 sig permission{}
3 abstract sig user {
4  ua:set role}
5 abstract sig role {
6 pa:set permission}
7 sig hierarchy{
8 rh:role -> role}
9 sig session {}
10 sig US {
11 us: user->session}
12 sig SR {
13 sr: session->role}
14 sig EXCLUSIVE {
15 xc:set role}
16 sig adminrole {
17 authority:set role}
18 one sig adminrelation {
19 can_assignp: adminrole -> role ->role,
20 can_revokep: adminrole -> role,
21 exigence: adminrole->obligation}
22 fact hierarchyfact{
23 all r1,r2: role, p: permission |r1 in r2.^(hierarchy.rh) and p in r1.pa implies p in r2.pa
24 all r1,r2: role |r1 in r2.^(hierarchy.rh)  implies r2 not in r1.^(hierarchy.rh)
25 no r: role |r in r.^(hierarchy.rh)  }
26 assert permissionassign { all r1,r2:role | r1 in r2.(hierarchy.rh) implies r1.pa+r2.pa=r2.pa}
27 fact modelfact {
28 all u: user |some r: role |u in ua.r
29 all r: role |some p: permission |r in pa.p
30 all r:role |some u:user |r in u.ua
31 all p:permission | some r:role |p in r.pa  }
32 fact DSD {all r1, r2:role |all s:session |
33 r1 in r2.xc implies ((s ->r1) not in SR.sr or (s->r2) not in SR.sr)}
34 fun oblg[u:user]:set obligation{u.ua}
35 fun assign_roletou[u:user]:set role {u.ua}
36 fun assign_roletop[p:permission]: set role{pa.p}
37 fact can_assign {all ar:adminrole,u:user,p:permission |
38 ar -> assign_roletop[p]-> assign_roletou[u] in adminrelation.can_assignp and
39 assign_roletou[u] in ar.authority and
40 p in assign_roletop[p].pa implies p in assign_roletou[u].pa}
41 pred PC_DSD (u:user,p:permission,s:session,exclusive:EXCLUSIVE)
42 {assign_roletou[u] in exclusive.xc and
43 assign_roletop[p]not in exclusive.xc and
44 (u->s) in US.us implies
45 (assign_roletou[u]->s) in SR.sr and
46 (assign_roletop[p]->s) in SR.sr }
47 run PC_DSD
48 assert exception {all s:session|all ar:adminrole |all u:user |all p:permission |
49 (u->s) in US.us and  ar->oblg[u] not in adminrelation.exigence
50 and ar->assign_roletop[p]->assign_roletou[u] in adminrelation.can_assignp
51 implies   assign_roletou[u] in assign_roletop[p].xc}
52 check exception
```

Figure 5. Specification of Exc-RBAC

The figure 5 shows the implementation of the formal specification in Alloy, we will start with the specification of the exception situation from line 48 to line 51, this code segment allows to check the correctness by using the assert feature in Alloy.Assert's role is to find a case that contradicts the predicate by showing the graph if there is a counterexample as shown in figure 6.

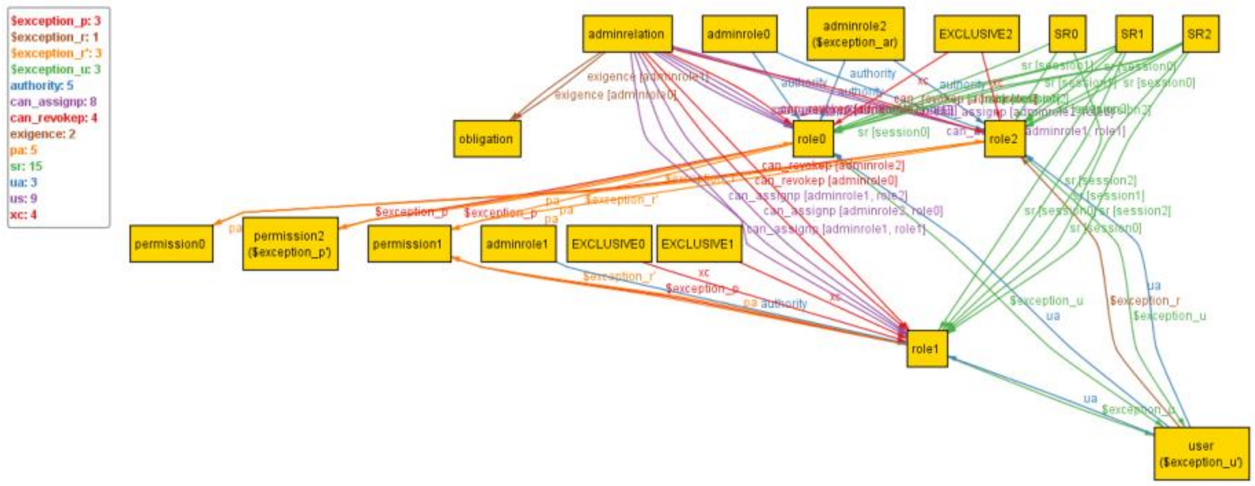
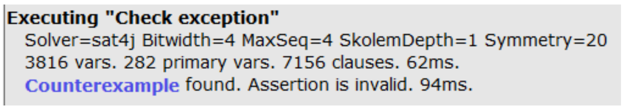


Figure 6. The found Counterexample

Executing command Check exception as shown in line 52 outputs the result shown in figure 7. This figure shows that “Assertion is invalid” and “Counterexample is found”. It took 94ms to determine invalidity and find a counterexample.



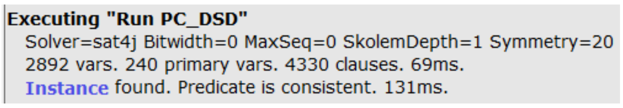
```
Executing "Check exception"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   3816 vars. 282 primary vars. 7156 clauses. 62ms.
   Counterexample found. Assertion is invalid. 94ms.
```

Figure 7. The result (when conditions are satisfied)

The permission is represented as a signature whose constituent elements are members of the set object and action as shown in line 2. Line 1 defines sig obligation. Line 3 defines sig user. In line 4, ua is used to assign role to users. Line 5 defines sig role, where pa is used to assign permission to a role as shown in line 6. Lines 7 and 8 define sig hierarchy role. Line 9 defines sig session. Lines 10, 11 define sig US, where US represents the correspondence between user and session. Lines 12, 13 define sig SR, where SR represents the correspondence between session and role. Lines 14, 15 define sig EXCLUSIVE. Line 16 defines sig adminrole, and line 17 specifies the authority of the adminrole on roles. In line 18, we define the sig adminrelation. This definition includes lines 19, 20 and 21 in which we specify the assignment of permission from one role to another, the revocation of a permission of a role and the execution of the permission.

In this specification, we use the notion of hierarchy in the form of constraints as shown from line 22 to 25. Line 26 defines assert permissionassign, assert defines constraints that the system must satisfy. By definition, a fact does not require arguments and specifies constraints that are applied on the signatures throughout the execution, for that fact has been used to define the model as shown from line 27 to line 31. Lines 32, 33 define the dynamic constraintes. In line 34, the obligation was introduced like function, oblg() return the obligations to execute by the user. Line 35 defines function assign_roletou() who returns the role of user. Line 36 defines function assign_roletop() who returns the role of permission. Lines 37, 38, 39 and 40 define fact can_assign, this code segment allows to execute constraints so that the permission is assigned (assign) to the role of the user.

The consistency test makes it possible to generate the instances, then validating them;this test is done by executing the predicates. To represent the dynamic constraint, we use the predicate function of Alloy, and can check if the role of the user and the role of the permission are conflicting in a session by using the Run command in Alloy analyzer.A graph is displayed if a valid entry is found by the analyzer. Lines 41-46 test the consistency of the system by declaring a predicate based on the defined functions.Figure 8 illustrates the "Run PC-DSD" execution.This figure shows the time it takes to determine consistency and find the instance.



```
Executing "Run PC_DSD"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   2892 vars. 240 primary vars. 4330 clauses. 69ms.
   Instance found. Predicate is consistent. 131ms.
```

Figure 8. Alloy Analyzer Output (checking consistency of dynamic constraints)

Figure 8 displays the instances by clicking on the instance link.

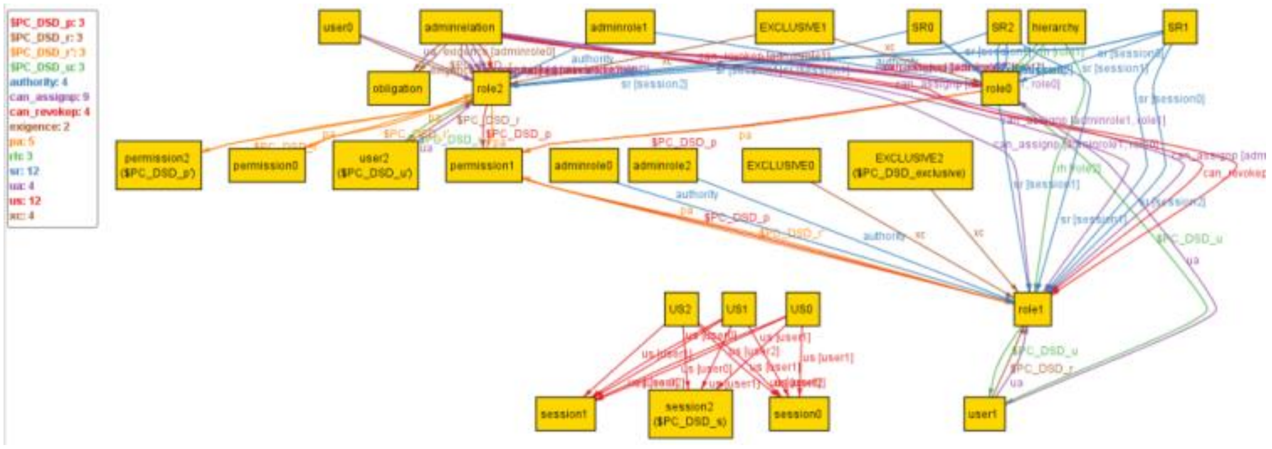


Figure 9. DSD Instances

6. An illustrative example (Clinical information system)

For the implementation of our model, we can look at examples of IS references such as: clinical information system and meeting scheduler system. In this study, we opted for the clinical information system as an illustrative example to demonstrate consistency with the security policy of clinical information system; this example is based on Anderson's paper [1].The clinical information system is based on the principle of hierarchy. The senior role inherits the junior role. In regular roles,permissions in the role hierarchy are inherited from the junior role from the senior role as illustrated in figure 10. The administrative roles hierarchy is presented in figure 11. Each administrative role manages a range of regular roles as shown in table 3. The corresponding users and roles are shown in table 1. Table 2 shows the associated permissions for each role. The main roles inherit permission from the junior role. For example, role RC inherits permissions from role C.

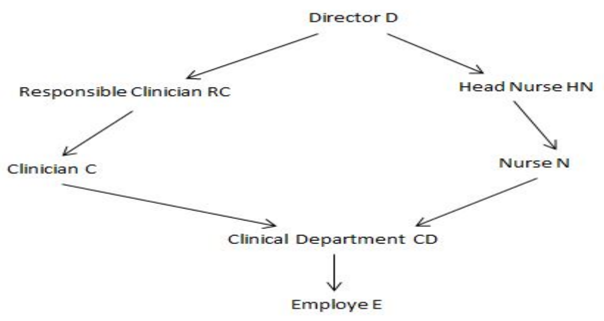


Figure 10. Regular Roles Hierarchy

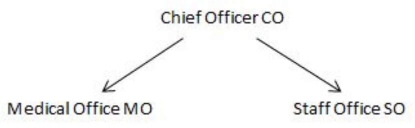


Figure 11. Administrative Roles Hierarchy

The proposed model applies to the clinical information system to ensure in addition to data confidentiality and record validity, the flexibility of access in exception situations. In this example, it is considered that each patient has a single medical record in which personal health information is recorded [10].

The system must restrict access to identified persons and deny access to others, these accesses are made on a medical record. The Responsible Clinician (RC) has wide privileges, he has the right to add other clinicians and must inform the patient of clinicians who have access privileges to the medical record at the time of opening the record. A clinician can open a record for a patient. In the case where a record is opened following a reference, the referring clinician must have the right of access to the record. The information in one record may be added to another record. The information cannot be removed from a record before the end of delay.

To better illustrate the example, we propose the following scenario: The user U2 is a doctor and plays the role C. Role C does not have the permission "P2: Notify the patient of the names of clinicians.Suppose user U2 needs in case of exception of permission P2. Thus, it puts the system in an exception situation. The system establishes a session. It calls the user U2 to execute obligations, such as "notify RC". Once all the obligations are fulfilled, the user requests P1 to the MO administrative role (the user U2 has the role C, and the role C is in the role range of the administrative role MO). So, MO use the can_assignp (MO, pc, C) relation in which pc is a precondition expressed as follows:

- Function assign_roletou(): returns role of user.
- Functionassign_roletop(): returns role with permission P.
- Before granting permission to the user's role, the union of the two roles() must not be dynamically contradictory (DSD), provided that pc is satisfied.

When the administrative role MO verified that there is no conflict in the dynamic DSD constraint.Then user U2 can have P2 permission.At the end of the emergency situation, a can_revokep(MO,C) relation is established, MO revokes the P2 permission of C and the session ends.

TABLE 1. USER-ROLE ASSIGNMENT

User	Role
U1	Responsible Clinician RC
U2	Clinician C
U3	Clinician C
U4	Patient P

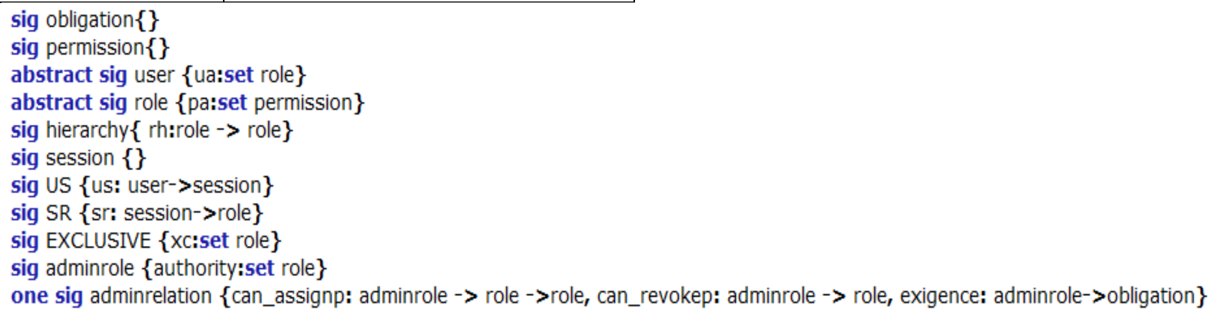
TABLE 2. ROLE-PERMISSION ASSIGNMENT

Role	Permission
Responsible Clinician RC	P1: Add clinicians P2: Notify the patient of the names of clinicians P3: Append P4: Read
Clinician C	P5: Append P6: Read
Patient P	P7: Read

TABLE 3. ADMINISTRATIVE ROLE RANGE

Administrator role	Role range
SO	[E,D]
MO	[CD,RC]
SO	[CD,HN]

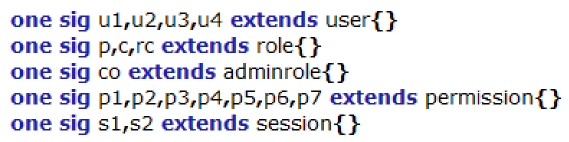
In the following, the example of the clinical information system will be translated into Alloy to evaluate the Exc-RB formal specifications. In the first phase, we give the domain model that represents the Exc-RBAC system as shown in figure12. Each entity will be represented by a set and the instances of the entities will be represented by the members of their corresponding set.



```
sig obligation{}
sig permission{}
abstract sig user {ua:set role}
abstract sig role {pa:set permission}
sig hierarchy{ rh:role -> role}
sig session {}
sig US {us: user->session}
sig SR {sr: session->role}
sig EXCLUSIVE {xc:set role}
sig adminrole {authority:set role}
one sig adminrelation {can_assignp: adminrole -> role ->role, can_revokep: adminrole -> role, exigence: adminrole->obligation}
```

Figure 12. Signatures specification

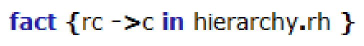
In the second phase, we use the inheritance concept to introduce the users of the clinical information system, the roles and the permissions associated with the roles, as shown in following figure 13.



```
one sig u1,u2,u3,u4 extends user{}
one sig p,c,rc extends role{}
one sig co extends adminrole{}
one sig p1,p2,p3,p4,p5,p6,p7 extends permission{}
one sig s1,s2 extends session{}
```

Figure 13. Example specification

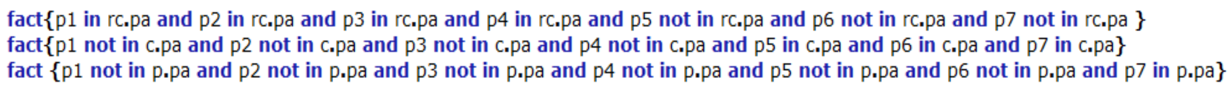
This model includes the hierarchical structure of the system. For regular roles, each RC oversees a set of clinicians as illustrated in figure 14. Similarly, for administrative roles CO oversees MO.



```
fact {rc ->c in hierarchy.rh }
```

Figure 14. Roles hierarchy

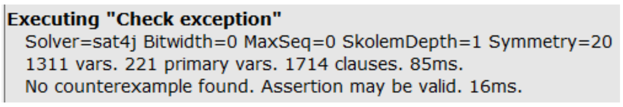
Permissions associated with roles are defined using the facts (figure15). Therefore, they are defined as conditions that are always true.



```
fact{p1 in rc.pa and p2 in rc.pa and p3 in rc.pa and p4 in rc.pa and p5 not in rc.pa and p6 not in rc.pa and p7 not in rc.pa }
fact{p1 not in c.pa and p2 not in c.pa and p3 not in c.pa and p4 not in c.pa and p5 in c.pa and p6 in c.pa and p7 in c.pa}
fact {p1 not in p.pa and p2 not in p.pa and p3 not in p.pa and p4 not in p.pa and p5 not in p.pa and p6 not in p.pa and p7 in p.pa}
```

Figure 15. Assignment

To check the exception, we use the check command. The execution produces the result illustrated in figure 16. This figure shows that the proposed case is valid, that is, in the current exception situation, the user U is granted the permission P.



```
Executing "Check exception"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  1311 vars. 221 primary vars. 1714 clauses. 85ms.
  No counterexample found. Assertion may be valid. 16ms.
```

Figure 16. Exception checking for the clinical information system

7. CONCLUSION

This article focuses on the exception situation which allows users to have unauthorized permissions in normal situations. This is done in order to increase the flexibility of RBAC model. The proposed model allows users to enrich roles with permissions in the active session. The separation of duties is at the core of the proposed approach. This dynamic separation aims at limiting and controlling the freedom of users in exception situations. The permission requested by the user must not cause a conflict between the roles in the current session.

This work provides an administrative model for managing users, roles, and permissions in large systems. Then, a formal specification of the proposed model has been validated by Alloy analyzer. In the end, Anderson's clinical information system has been proposed as an illustration with the validity check. The extension of this work will be devoted to proposition a tool that allows to integrate SecureUML and Alloy. The SecureUML and Alloy specifications are two different views of the system. For this, an MDA transformation will be used.

REFERENCES

[1] Anderson, R. A., "Security Policy Model for Clinical Information Systems, " In Proceedings of the 1996 IEEE Symposium and Security and Privacy, Oakland, USA: IEEE Press, pp. 30-43, 1996.

[2] ANSI INCTS (ed), American National standard for Information Technology. Role-based access control (INCITS 359-2004). New York, USA : ANSI, Inc, 2004.

[3] Benedetti, M., & Mori, M., "Parametric RBAC Maintenance via Max-SAT," 23rd Symposium on Access control Models and Technologies, pp.15-25, Indianapolis, USA, 2018

[4] Bertino, E., Bonatti, P. A., & Ferrari, E., "TRBAC: a temporal role-based access control model," ACM Transactions on Information and System Security, pp.191-233, 2001. doi: 10.1145/501978.501979.

[5] Cotrini, C., Weghorn, T., Basin, D., & Clavel, M. "Analyzing First-order Role Based Access Control," 28th Computer Security Foundations Symposium, pp.3-17, 2015.

[6] Crampton, J., & Khambhammettu, H. "On delegation and workflow execution models," In Proceedings of the 2008 ACM symposium on Applied computing, pp. 2137-2144, 2008.

[7] Cuppens, F., Cuppens-Boulahia, N., BenGhorbel-Talbi, M., Morucci, S., & Essaouni, N. (2013). "Smatch: Formal dynamic session management model for RBAC," Journal of information security and applications, vol.18, pp.30-44, 2013.

[8] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. "Proposed NIST standard for role-based access control," ACM Transactions on Information and System Security, pp.224-274, 2001. doi: 10.1145/501978.501980.

[9] Ferreira, A., Chadwick, D., Farinha, P., & Cruz-Correia, R. J. "How to security break into RBAC: the BTG-RBAC model," Computer security applications conference, pp.23-31, 2009. doi: 10. 1109/ACSAC.2009.12

[10] Haraty, R. A., & Naous, M. "Modeling and validating the Clinical Information Systems Policy using Alloy," In Proceedings of the Second International Conference on Health Information Science, London : Lecture Notes in Computer Science-Springer, pp.1-17, 2013.

[11] Illand, J. Politique de sécurité des systèmes d'information. Sécurité Informatique. Paris, France. Centre National de la Recherche Scientifique, 2006.

[12] Jackson, D., Schechter, I., Shlyahter, H. "Alcoa: the alloy constraint analyzer," In Proceedings of the 22nd. International Conference on Software Engineering. Ireland: ACM Press, 2000.

[13] Jackson,D. Software abstraction: logic, language, and analysis. Cambridge, USA: MIT Press, 2006.

[14] Jonscher, D. Extending access controls with duties realized by active mechanisms, Database Security VI:Status and Prospects, pp. 91-112, North-Holland : Elsivier, 1993.

[15] Logstaff, J. J., Lockyer, M. A., & Thick, M. G. "A model of accountability, confidentiality and override for healthcare and other applications," In Proceedings of the fifth ACM workshop on role-based access control, pp. 71-76, 2000. doi: 10.1145/ 344287.344304

[16] Liu, G., Zhang, R., Song, H., Wang, C., Liu, J, & Liu, A. "TS-RBAC: a RBAC model with transformation," Computer and Security, vol.60, pp. 52-61, 2016.

[17] Lu, J., Xin, Y., Zhang, Z., Peng, H., & Han, J. "Supporting user authorization queries in RBAC systems by role–permission reassignment," Future Generation Computer Systems, vol.88, pp.707-717, 2018.

[18] Maw, H. A., Xiao, H., Christianson, B., & Malcolm, J. A. "An evaluation of break-the-glass access control model for medical data in wireless sensor networks," IEEE 16th international conference on e-health networking, applications and services, 2014.

[19] Nazerian, F., Motameni, H., & Nematzadeh, H. "Emergency role-based access control (E-RBAC) and analysis of model specifications with alloy," Journal of Information Security and Applications, vol.45, pp. 131-142, 2019.

[20] Ni, Q., Bertino, E., & Lobo, J. "An Obligation Model Bridging Access Control Policies and Privacy Policies," In Proceedings of the 13th ACM symposium on Access control models and technologies (SACMAT '08). NY, United States : Association for Computing Machinery, 2008.

[21] Oh, S., & Sandhu, R. "A model for role administration using organization structure," In Proceedings of the seventh ACM symposium on Access control models and technologies (SACMAT '02). NY, United States : Association for Computing Machinery, 2002.

[22] Oh, S., Sandhu, R., & Xinwen, Z. An effective role administration model using organization structure. ACM Transactions on Information and System Security, vol.9, no.2, pp.113-37, 2006.

[23] Osborn, S. L., & Wang, H. "A survey of delegation from an RBAC perspective". Journal of Software, vol.8, no.2,pp. 266–275, 2013.

[24] Pontual, M., Chowdhury, O., Winsborough, W, H., Yu, T., & Irwin, K. "On the Management of User Obligations," In Proceedings of the 16th ACM symposium on Access control models and technologies (SACMAT '11), Innsbruck, Austria, 2011.

[25] Rissanen, E., Firozabadi, B. S., & Sergot, M. "Towards a mechanism for discretionary overriding of access control," In Proceedings of the 12th international workshop on security protocols, 2004.

[26] Sandhu, R., Coyne, E., Feinstein, H., & Youman, C. (1996). "Role-Based Access Control Models," IEEE Computer, vol.29, no.2, pp.38–47, 1996.

[27] Sandhu, R., Bhamidipati, V., & Munawer, Q. "The ARBAC97 model for role-based administration of roles," ACM Transactions on Information and System Security, vol.2, no.1, pp. 105-135, 1999.

[28] Sandhu, R., & Munawer, Q. "The ARBAC99 model for administration of roles," In Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC '99), DC,United States : IEEE Computer Society, 1999.

[29] Schaad, A., Moffett, J. D., & Jacob, J. "The role-based access control system of a European bank: a case study and discussion," In Proceedings of the 6th ACM symposium on access control models and technologies, pp. 3-9, 2001.

[30] Schaad, A., & Moffett, J. D. "Delegation of obligations," In Proceedings of the 3rd international workshop on policies for distributed systems and networks, pp. 25-35, 2002.

[31] Schefer-Wenzl, S., & Strembeck, M. "Generic support for RBAC break-glass policies in process-aware information systems," In Proceeding of the 28th ACM symposium on applied computing, pp. 1441-1446, 2013.

[32] Schefer-Wenzl, S., Bukvova, H., & Strembeck, M. "A review of delegation and break–glass models for flexible access control management," International conference on business information systems, pp. 93–104, 2014.

[33] Sharma, M., Sural, S., Vaidya, J., & Atluri, Y. "AMTRAC: an administrative model for temporal role-based access control," Computers & Security, vol.39, pp.201-218, 2013.

[34] Sohr, K., Kuhlmann, M., Gogolla, M., Hu,H., & Ahn,G.J. "Comprehensive two-level analysis of role-based delegation and revocation policies with UML and OCL," Information and Software Technology, vol.54, no.12, pp. 1396-1417, 2012. doi: 10.1016/j.infsof.2012.06.008.

[35] Subramanian, C. M., Cherukuri, A. K., & Chelliah, C. "Role based access control design using three-way formal concept analysis," International Journal of Machine Learning and Cybernetics, vol.9, pp. 1807-1837, 2018.

[36] Vaziri, M., Vaziri, A., & Jackson, D. "Some Shortcomings of OCL, the Object Constraint Language of UML", In Proceedings of the Technology of Object-Oriented Languages and Systems TOOLS '00, 1999.

[37] Wainer, J., Barthelmess, P., & Kumar, A. "W-RBAC - a workflow security model incorporating controlled overriding of constraints," International Journal of Cooperative Information Systems, vol.12, no.4, pp.455-485, 2003. doi: 10.1142/S02188430 030 0 0814.

[38] Zhang, Y., & BDJoshi, J. "ARBAC07: a role-based administration model for RBAC with hybrid hierarchy," IEEE international conference, IL, USA, 2007. doi: 10.1109/IRI. 2007.4296620.

[39] Zhao, G., Chadwick, D., & Otenko, S. (2007). "Obligations for role-based access control," 21st International Conference on Advanced Information Networking and Applications Workshops, pp. 424-431, 2007. doi: 10.1109/AINAW.2007.267.

A.BOUADJEMI Ph.D. student in the Computer Science at the Oran 1 University, Algeria. His research interests include information systems security, formal specification and information systems.

M.K. ABDIholds a master degree and a PhD degree in computer science from Department of Computer Science at the Oran 1 University, Algeria. He is currently, professor for the same Department, and researcher in RIIR laboratory. His research interests include the application of Artificial Intelligence techniques to Software Engineering, software quality, software evolution, formal specification, Systems analysis and simulations, Data-Mining and Information Systems.