ADV MATH SCI JOURNAL
Advances in Mathematics: Scientific Journal **12** (2023), no.3, 465–477
ISSN: 1857-8365 (printed); 1857-8438 (electronic)
https://doi.org/10.37418/amsj.12.3.4

FORMAL DECISION MODELING FOR ROLE-BASED ACCESS CONTROL POLICIES

Bouadjemi Abbdelkrim

ABSTRACT. Role-Based Access Control (RBAC) has been widely used in information systems, including so-called critical systems. In business, workflows are used to control the flow of processes. One of the major issues concerning these processes is to be able to verify that a proposed process model strictly corresponds to the specifications to which it is supposed to respond. Access control models describe the frameworks that dictate permissions. The RBAC model is generally static, i.e. the access control decisions are: grant or deny. Dynamic and flexible access control is required. In order to increase the flexibility of access control, the notion of decision has been proposed. Decisions execute the requirements to be fulfilled.

The main of this article is to use the decision to produce a dynamic model. Our model augments the dynamics of the RBAC model. It allows dynamically assigning permissions. For illustration, Feather's meeting management system is used. Finally, first-order logic is used to analyze the validity of the proposed model.

1. INTRODUCTION

With the growth of information systems (IS), the design and modeling of functional aspects without taking into account the security part exposes organizations

2020 *Mathematics Subject Classification.* 20F10, 03B10, 68M01, 68M04.
Key words and phrases. formal specification, first-order logic, decision, access control, RBAC, SecureUML.
Submitted: 29.01.2023; *Accepted:* 14.02.2023; *Published:* 18.03.2023.

to risks. Faced with the risks incurred, it is necessary to identify what must be protected, to quantify the corresponding issue, to formulate security objectives and to identify and implement the provisions adapted to the right level of security retained. This primarily involves defining and implementing an access control policy.

Access control is an essential element of IS protection. Most organizations define roles for different organizational tasks. RBAC is the most suitable for these organizations and the most used for the IS in terms of access control. In 1996, Sandhu et al. in [1] proposed role-based access control.

Semi-formal IS approaches such as UML (Unified Modeling Language) and SecureUML have shown their limits. They do not verify or validate the specified properties and constraints. The ability to analyze semi-formal models and reason about properties of a system early in the development process has received considerable attention. Formal specification methods are used in software engineering to reason about mathematical models. The interest is to prove or verify properties on these models. Despite the overhead associated with analysis and design tasks in formal specifications, the use of such methods is increasingly justified for software that involves critical data or security conditions. One of the difficulties is to formalize the information systems. This work proposes to improve the flexibility of access control to allow the IS to adapt to different situations, i.e. a role that does not have a given permission can request it in a given situation. unexpected but under certain conditions. The proposed model makes it possible to authorize a rejection decision. For illustrative purposes, we've used Feather's meeting management system as an example [2]. Next, the model and example specifications were modeled by first-order logic. It is used for specifying and verifying access control policies.

In the rest of the article, Section 2 reviews the literature on the issue. Section 3 presents the RBAC model as well as the SecureUML metamodel. In section 4, we start by presenting Feather's scheuling organization system, then show the functional and security model. In section 5, the modeling approach is presented. In Section 6, the sample access control decision illustrates this work. Finally, the last section ends the article and presents some perspectives.

2. RELATED WORK

Consideration of access control policies is a major concern. Among the types of access control, the most widely used is RBAC. Chen and Sandhu [3] proposed a formal language based on the first-order language to express constraints. In [4], the authors formally specify a model called Smatch (Secure MAnagement of swiTCH). In the model, the dynamic session is created by the user. 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 the queries to the question of Satisfiability Modulo Theories (SMT). In [6], authors proposed Emergency RBAC (E-RBAC) whose purpose is to enhance flexibility, this approach uses BTG policy and separation of duties (SOD) is included whose purpose is to control user access in the event of an emergency. Alloy was used to implement the model specifications and the medical scenario. In [7], the authors proposed Exc-RBAC, the goal is to assign permissions dynamically by increasing the flexibility of the RBAC model in order to handle exception situations. As an illustration, Anderson's clinical information system is used. Finally, Alloy is used to analyze the validity of the proposed model. The idea in this article is the reassignment of permissions to roles.

Chehida et al [8] who presented an approach combining UML and B for the specification of RBAC policies. They modeled the rules of the rules which are then translated into formal specifications thanks to B4Msecure. In [9], the authors developed a tool to analyze RBAC security models. They used a transformation approach in SecureUML2Prolog to transform SecureUML models into Prolog. In [10], the authors proposed an approach for the formal development of a secure filter that regulates access to the IS. They developed a tool to automate the following steps: the modeling of system aspects by class diagrams, SecureUML for security rules and activity diagrams for dynamics. A set of rules used for the generation of a B specification. Finally, the B implementation is translated into the AspectJ implementation.

Shaikh in [11] proposes two risk-based dynamic decision methods for access control systems, they provided theoretical and simulation-based analysis and evaluation of the two schemes. They also analytically showed that the proposed methods not only allow exceptions under certain controlled conditions, but uniquely

limit legitimate access for authorized users. The approach proposed in this article is based on the concept of decision, which allows to order an unauthorized access of an authorization on the basis of the requirements.

3. THE RBAC MODEL AND THE METAMODEL SECUREUML

Roles are used as basis constructors in access control model by Sandhu et al. in [1]. The basic model, the hierarchical model, the constraint model, and the consolidated model are the four models for RBAC. Users, roles, operations, objects, and sessions compose the five fundamental components of Core RBAC (American National standard for Information Technology, 2004), with five relationships: user assignment (UA), permission assignment (PA), user session (US), session roles (SR), and permissions. The model is shown in Figure 1 below:

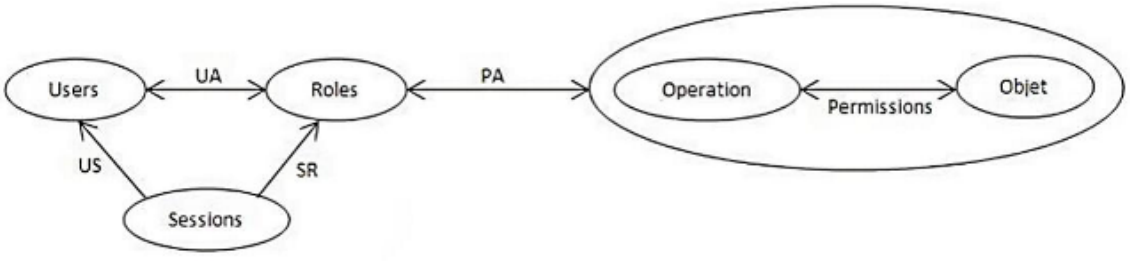

FIGURE 1. RBAC model

UML [12] is a standard graphical notation of the OMG (Object Management Group) used for the analysis of requirements and the design of a system. The concept of "profile" allows the extension of UML diagrams to specify a particular aspect of a system. Several works have proposed useful profiles for the specification of access control policies based on the RBAC model. Among these profiles we can cite AuthUML [13] which offers extensions of the use case diagram and UMLsec [14] which presents a profile to extend the activity diagram.

In our approach, we use the SecureUML profile [15] [16] which allows to represent a security policy based on the RBAC model in a static view. This diagram uses permissions, represented by associative classes, to express access control rules. A permission is linked to a class stereotyped by “Role” which represents the users assigned to the role, and another class stereotyped by “Entity” which represents

the target class of the permission. The permission thus specifies the access rights of users to the entity it protects. It is defined by a set of attributes indicating the types of authorized actions (reading, modification, etc.).

These attributes are defined by three properties: stereotypes to specify the type of the resource to be protected, the name of the attribute which determines the protected resource, and the type of the attribute to specify the action authorized by the permission. It is possible to subject this permission to contextual conditions, called authorization constraints.

4. MEETING SCHEDULING SYSTEM

Our approach is based on Feather's example of scheduling meetings [2], this example is as follows:
Initiator organizes a meeting; for this, he must invite participants, find a place (local) and the moment (time) of the meeting. As the meeting must be secret, the meeting management system must apply a security policy, i.e., the time and place could be introduced and modified only by the Initiator, and could be consulted only by the participants "Participant" (invited to the meeting).

For this, we have drawn up the following set of rules:

- All system users are allowed to create new meetings and read all meeting entries.
- Only the Initiator of a meeting is authorized to modify the meeting data and to cancel or delete the meeting.
- The Initiator is authorized to cancel all meetings.
- A Participant can also read this information.
- Initiator can create, read, delete and update invitations to these meetings.
- A participant can read and respond to invitations.
- A participant can create a proposal change for a meeting following an invitation and update this change.
- Initiator can read and respond to a change associated with an invitation of his meeting.

In this section, we present the UML models of a meeting management system.

4.1. **Functional model.** The functional model in Figure 2 describes the system specifications without taking authorization constraints into account. It defines the different requirements at the functional level:

- Initiator cannot create two meetings in one time.
- Only the initiator can create meeting invitations.
- A meeting is associated with a single initiator.

Initiator can be a participant in another meeting that takes place in another time.

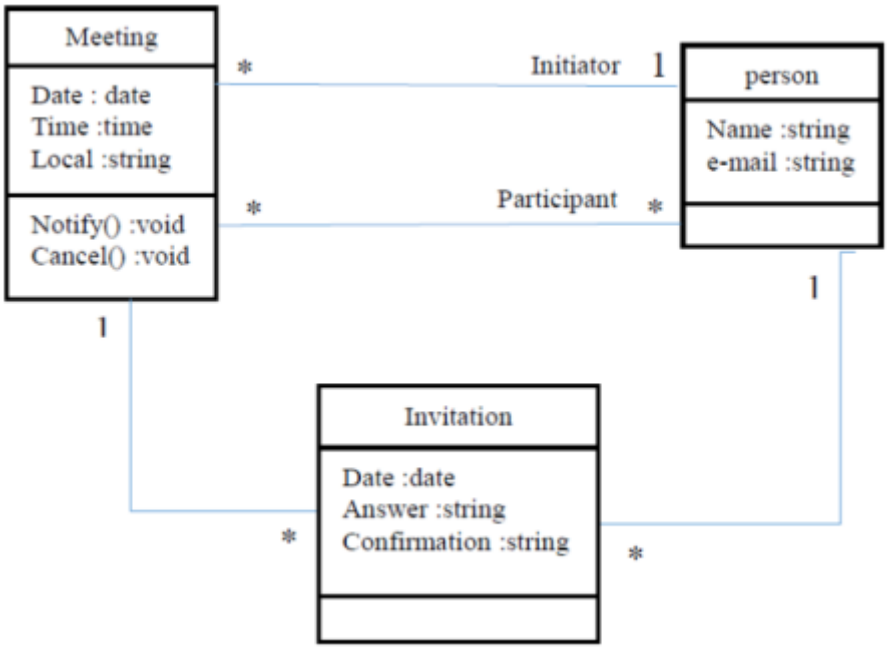

FIGURE 2. functional model

4.2. **Security Model.** The security model defines an access control policy based on the roles of the various users of the system and assigns the roles to the users.

In our case study, there are two roles: Initiator and Participant. To each of these roles, we assign several permissions that activate operations acting on the functional classes.

To simplify the model, we do not consider the dynamic separation of duties (Initiator and Participant are conflicting).

A user who has the Initiator role:

- Has full read and write access to meeting info.

- Is authorized to cancel all meetings.
- Has full read and write access to information related to invitations.

A user who has the Participant role:

- Can read this info.
- Can read and respond to invitations.
- Can create proposal changes for a meeting.

The SecureUML meta-model based on the RBAC model presents concepts such as user, role and permission as well as the relationships between them.

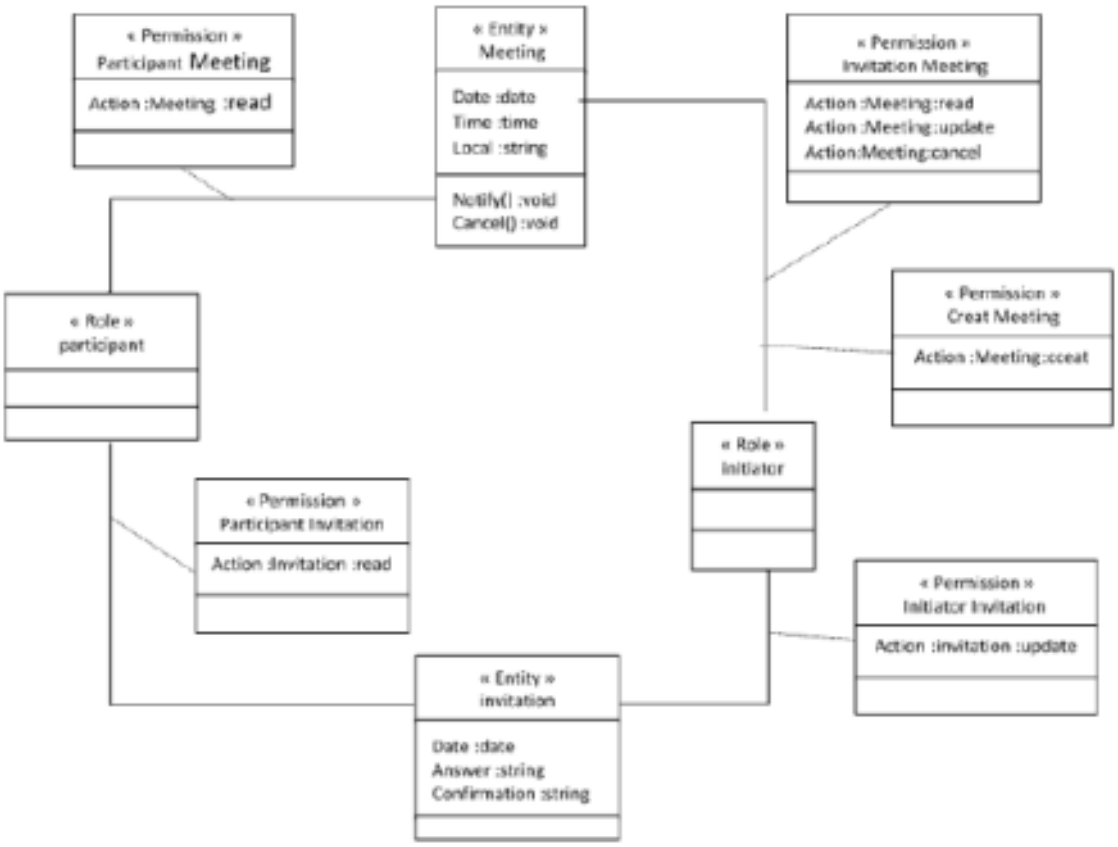

FIGURE 3. SecureUML model of access control for meeting scheduler example

Figure 3 shows the RBAC pattern on the meeting scheduler example. In order to complete the modeling, we will introduce the authorization constraints expressed in OCL as follows:

IM.AC Caller=Self.creator.name
PM.AC Self.invitation.person → exists (Caller=name)
II.AC Self.meeting.creator.name=Caller
PI.AC Self.person.name=Caller

5. Modeling approach

Our security policy is based on the notion of permission. This may be permitted or prohibited. From there, we can define the security policy as follows.

A security policy defined on a set of actions is a property indicating for each action whether it is authorized or not.

In our study, the following definition was given to the security policy: A system corresponds to a security policy if any system event is linked to an authorized action of the policy.

The constraint when must always guarantee is that the security policy must be consistent, that is, it must be shown that the set of rules constituting the security policy does not contain any conflicting requirements.

5.1. **Modeling choice.** At the end of the development, we must have a secure system based on RBAC decisions. SecureUML formalizes access control decisions that depend on two types of information:

- Static access control decisions that depend on static information, namely the assignment of Users and Permissions to Roles, designated by the RBAC configuration;
- Dynamic access control decisions that depend on dynamic information, i.e., the satisfaction of permission constraints in the current state of the system.

Using first-order logic, we can formulate static decisions as:

- RBAC configuration as CRBAC first order structure;
- Semantics of static access control decisions $C_RBAC \models \phi_RBAC(u,a)(u,a)$.

Formulate dynamic decisions as:

- State system presented by first-order structure;
- Permission constraints as a first-order formula.

To be able to formulate the semantics of individual access control decisions, one must combine RBAC configuration and authorization constraints. SecureUML is used to represent the system.

5.1.1. *Static access control.*

$AC \subseteq users * permissions$

$AC \in UA.PARelationdecomposition$

$AC \subseteq usersroles$

We define the signature $\sum_{RBAC}(S_{RBAC} \leq F_{RBAC}P_{RBAC})$ which defines the type of structure specifying the RBAC configuration. We have:

- S_{RBAC} Is a set of Lattices,
- $\leq_{RBAC}$ Is a partial order in S_RBAC,
- F_{RBAC} Is a lattice set of functions,
- P_{RBAC} Is a lattice set of predicates.

5.1.2. *Lattice definition.* Let E be a set and R a binary relation on E, to show that (E,R) is a Lattice, it is necessary that:

- (E,R) is an ordered set
- Then show that

$$S_{RBAC} = Users, Subjects, Roles, Permissions, AtomicActions, Action$$

$$\begin{cases} users \leq_{RBAC} subjects \\ AtomicsActions \leq_R BACActions \end{cases}$$

$$\begin{cases} \geq_{RBAC}: subject * subject.UA : subjectsroles \\ \geq_{roles} \geq = roles * roles.PA : rolespermissions \\ \geq_{Actions}: Actions * Actions.AA : PermissionsActions \end{cases}$$

We define the partial order $\leq_{roles}$ in the RoleHierarchy aggregation association in Role, and we write that the subrole "subrole" (the role with additional privileges) to the left of symbol. The partial order $\leq_{Actions}$ is defined by a reflexive closure of the hierarchical composition association in actions, defined by the aggregation ActionHierarchy.

A binary relation R on E is a relation of partial order if and only if:

- R is reflexive
- R is transitive
- R is antisymmetric

We write $a_1 \geq a_2$, if a_2 is the subordinate action of a_1. Then, the formula $\phi_R BAC(u, a)$ is:

$$\phi_R BAC(u, a) = \exists s \in subject, r_1, r_2 \in role, p \in permission, a_2 \in Action,$$

$$S \geq_s ubjectu \wedge UA(s, r_1) \wedge r_1 \geq_r oler_2 \wedge PA(r_2, p) \wedge AA(p, a_2) \wedge a_2 \geq_A ctiona_1.$$

We can factorize the formula according to the definition of access control:

$$AC \doteq PA.UA$$

$$\phi_R BAC(u, a) = \phi_u sers(u, p) \wedge \phi_A ction(p, a)$$

Such as: $\phi_u sers(u, p) \doteq \exists s \in subject, r_1, r_2 \in role$

$$s \geq_s ubjectu \wedge UA(s, r_1) \wedge r_1 \geq_r oler_2 \wedge a_1 \geq_A ctiona_1$$

At the end, the static access control is defined as: User u must execute an action a if $C_R BAC \models \phi_R BAC(u, a)valid$.

5.1.3. *Dynamic access control.* Static access control decisions related to models and not to system behaviors, for this, we must incorporate these decisions with dynamic decisions that act on the behavior of the system. Formally, we gave the signature $\sum_S T(S_S T, F_S T, P_S T)$ Such as:

$S_S T$ Trellis for classes (each class)

$F_S T$ Function for each attribute and for each method of the model

$P_S T$ Predicates for relations between classes.

However, we require that: $S_S T$ Contains the Users Trellis, and $F_S T$ contains Caller from the Users Trellis, and Selfc for each C class. Here we require that Selfc be interpreted by the current access to the object. When the current access to the object is from Trellis C, and Caller is interpreted by User who initiated this access. In this context, the state of the system at a particular time defines $\sum_S Tstructure\sigma_S T$, the constraints on the state of the system $\sigma_S T$ can be expressed as the logical formula $\phi_S T$, or the satisfaction of the constraints is:$\sigma_S T \models \phi_S Tvalid$.

5.2. **Sample access control decision.**

Atomic Actions ={Meeting: notify, cancel. Invitation: ...}

Actions = {Atomic Action u {Meeting,cancel,...}}

UA ={(Alice, Initiator),(Bob, Participant)}

PA = {(Iniator,creat Meeting) ...}

AA ={(Invitation.Meeting, Meeting: notify),(Initiator.Invitation, Invitation: ...)}

S = {Meeting, Invitation, Person}

P = {Meeting Initiator, Meeting Participant}

The constant $Self_Meeting$ of the Meeting Trellis indicates the current access to the Meeting. In this scenario, we will assume that Alice wants to cancel a Meeting initiated by Joe, the state of the system is a first order structure

$\sigma_{ST} on \Sigma_{ST}$

Caller $\sigma_{ST} = Alice$

Meeting $\sigma_S T = \{MeetingJoe\}$

Person $\sigma_S T = \{Alice, Joe\}$.

UserName $\sigma_S T = \{(Alice; "Alice"); (Joe, "Joe")\}$

Meeting Initiator $\sigma_S T = \{MeetingJoe, Joe\}$

Self $\sigma_S T = MeetingJoe$

The formula that must be satisfied by the structure $\sigma_A C = (\sigma_R BAC, \sigma_S T)$ to grant access to Alice according to: $\phi_A C(u, a) = \phi_u ser(u, p) \wedge \phi_A ction(p, a) \wedge \phi_S T^P(u)$. In the example, Alice has the Initiator Meeting permission to execute the action: Meeting:cancel(), so this action has heavy consequences on Alice; for this, Alice executes the Meeting:updat() action, because the Initiator role inherits from the Participant role.

For this action, no other action is possible, so the formula $\phi_u ser(Alice, p) \wedge \phi_A ction$ (p,Meeting:cancel) is true for this permission.

The constraint: Caller.name=Self.Participant.name on the Participant Meeting permission is translated into the formula:

$$UserName(Caller) = PersonName(MeetingParticipant(Self_Meeting()))$$

6. CONCLUSION

This article focuses on reassigning unauthorized permissions. This is done in order to increase the dynamics of the RBAC model. The proposed model allows users to enrich roles with additive permissions. The authorization requested by the user must not cause a conflict between the roles. Our approach allows access decision-making following the dynamics of the system. Then, a formal specification of the model was proposed by first-order logic. In the end, the meeting management

system was proposed as an illustration with the validity check. The extension of this work will be devoted to the proposal of a decision support system.

REFERENCES

[1] R.S. SANDHU, E.J. COYNE, H.L. FEINSTEIN, C.E. YOUMAN: *Role-Based Access Control Models*, (1996), p.22.

[2] M.S. FEATHER, S. FICKAS, A. FINKELSTEIN, A. VAN LAMSWEERDE: *Requirements and Specification Exemplars*, Automated Software Engineering, **4** (1997), 419–438.

[3] F. CHEN, R.S. SANDHU: *Constraints for role-based access control*, in Proceedings of the first ACM Workshop on Role-based access control - RBAC '95, Gaithersburg, Maryland, United States, (1996), p.14.

[4] N. CUPPENS-BOULAHIA, F. CUPPENS, M. NUADI: *Smatch Model: Extending RBAC Sessions in Virtualization Environment*, in Proceedings of the 2011 Sixth International Conference on Availability, Reliability and Security, USA, (2011), 17-26.

[5] C. COTRINI, T. WEGHORN, D. BASIN, M. CLAVEL: *Analyzing First-Order Role Based Access Control*, in 2015 IEEE 28th Computer Security Foundations Symposium, Verona, (2015), 3-17.

[6] F. NAZERIAN, H. MOTAMENI, H. NEMATZADEH: *Emergency role-based access control (E-RBAC) and analysis of model specifications with alloy*, Journal of Information Security and Applications, **45** (2019), 131-142.

[7] A. BOUADJEMI, M.K. ABDI: *Towards An Extension Of RBAC Model*, IJCDS, **10**(1) (2021), 1145-1155.

[8] S. CHEHIDA, A. IDANI, Y. LEDRU, M.K. RAHMOUNI: *Extensions du diagramme d'activité pour la spécification de politiques RBAC*, Ingénierie des systèmes d'information, **21**(2) (2016), 11-37.

[9] M. H. ALALFI, J. R. CORDY, T. R. DEAN: *Automated verification of role-based access control security models recovered from dynamic web applications*, in 2012 14th IEEE International Symposium on Web Systems Evolution (WSE), (2012), 1-10.

[10] A. MAMMAR, T. M. NGUYEN, R. LALEAU: *A formal approach to derive an aspect oriented programming-based implementation of a secure access control filter*, Information and Software Technology, **92** (2017), 158-178.

[11] R. A. SHAIKH, K. ADI, L. LOGRIPPO: *Dynamic risk-based decision mhods for access control systems*, Comput. Secur., **31**(4) (2012), 447-464.

[12] OMG UNIFIED MODELING LANGUAGE SPECIFICATION – VERSION 1.5, 2003. https://glossar.hs-augsburg.de (2022).

[13] K. ALGHATHBAR: *Representing Access Control Policies in Use Cases*, International Arab Journal of Information Technology, **9**(3) (2010), 268-275.

[14] J. JÜRJENS P. SHABALIN: *Automated Verification of UMLsec Models for Security Requirements*, in The Unified Modeling Language. Modeling Languages and Applications, Berlin, Heidelberg, 2004, 365-379.

[15] D. BASIN, J. DOSER, T. LODDERSTEDT: *Model driven security: From UML models to access control infrastructures*, ACM Trans. Softw. Eng. Mhodol., **15**(1) (2006), 39-91.

[16] D. BASIN, M. CLAVEL, J. DOSER, M. EGEA: *Automated analysis of security-design models*, Information and Software Technology, **51**(5) (2009), 815-831.

[17] A. BOUADJEMI, M. RAHMOUNI: *Modélisation formelle pour la sécurité d'un système d'information*. 2012.

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF RELIZANE
ALGERIA.
Email address: abdelkrimbouadjemi@yahoo.fr