International Journal of Control, Energy and Electrical Engineering (CEEE)
Vol.7 pp.12-17

A Correct-by-Design Role-Based Access Control Model for Healthcare Information Systems

Zoubeyr Farah*,1, Hania Gadouche*,2, Abdelkamel Tari*,3
*LIMED Laboratory, Department of Computer Science
Faculty of Exact Sciences
Abderrahmane Mira University, Bejaia, Algeria
Email: 1zoubeyr.farah@gmail.com 2gad.hania@gmail.com 3tarikamel59@gmail.com

***Abstract*—In this paper, a correcte-by-design approach is proposed to specify Role-Based Access Control (RBAC) for Healthcare Information Systems. The proposed approach is based on the Event-B formal method and its tool support RODIN plateform. To design a valid and multilevel access control model, a number of refinement operations are performed leading to a specification with several abstraction levels, each level implements selected properties according to the RBAC standard.**

***Index Terms*—Correct-by-Design, Event-B, Formal Methods, Healthcare Information System, Proof and Refinement, Role-Based Access Control, Specification and Validation.**

I. Introduction

In Healthcare Information Systems (HIS), wrong access definitions can lead to the violation of patient privacy due to the sensitive nature of the handled data. Access Control (AC) is a solution that is commonly used for controlling access to resources. Role-Based Access Control (RBAC) is currently the most used model as the management of access control is simplified; because defined permissions are granted to roles. In fact, RBAC is an ANSI standard [1], [2] that is used principally in industry [3] and enterprise management systems. An access control policy specification must be validated before its deployment. The specification validation includes the validity of the declarative aspect (i.e. properties definition) and the correctness of the behavioral one. The declarative aspect of RBAC considers the typing and the definition of relations between its components. The correctness of the behavioral aspect implies the preservation of the specification consistency. For this purpose, many methods are defined in the literature and several specification models for RBAC have been proposed [4]. Among the different formalisms used to model access control: SPIN model checker [5], LTL (Linear Temporal Logic) formalism [6], Alloy [7] CPN-Tool for Colored Petri Nets [8] and EB3SEC [9]. Regardless of used formalism, validation methods are generally based on a-posteriori verification process [10]–[12], or on a combination of both a-posteriori and a-priori verification process [13]–[15]. However, the increasing complexity of systems makes the a-posteriori verification process more difficult. To overcome this limit, some works have used formal methods to define a full-based a-priori verification process [16], [17] . Specification approaches that are based on formal methods, such as Event-B [18], are widely used to validate faultless critical systems. The use of formal methods requires understanding the system behavior and the relations between its components. Several works in the literature have used the Event-B method to specify the RBAC standard. In [19], access control is modeled in many levels of abstraction where each level defines a class of properties, as well, a two-step refinement approach is proposed to generate the model. In [12], an Event-B formulation of an E-Marking System access control is proposed. The verification and validation of the proposed formulation are carried out using the ProB tool [20]. Authors in [17] propose to validate the 2012 RBAC standard using the B method [21]. Most existing formal validation approaches deal only with the declarative aspect of the model without covering the behavioral features. In this paper, we deal with both of the declarative and the behavioral aspects. We use the Event-B formal method, since it allows the specification of systems according to a correct-by-construction methodology, additionally, it provides a large selection of tools and techniques for specifying, validating and checking properties of systems. Our approach is suitable for healthcare information systems where the validation of the constructed specification is crucial. The proposed specification approach is based on refinement and gives a multilevel view of the access control model. The refinement starts from a high abstract level of specification to the most concrete one. The specification consistency is preserved through all its levels as the model properties are linked to the behavior. Indeed, defining properties of the model according to its behavior allows to get an a-priori verification of the specification correctness. Our approach is based on an a-priori formal verification process, similarly to the one proposed to validate communicating systems in [22], and the one applied on Attribute-based access

ISSN 2356-5608

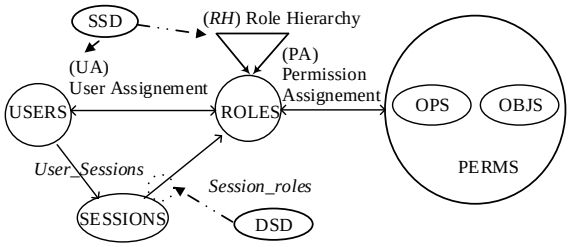

Fig. 1. RBAC model

control (ABAC) [23], [24] . RODIN platform [25] is used to develop and validate the model specification. The remainder of the paper is organized as follows: Section 2 gives a presentation of the RBAC standard. Section 3 describes the Event-B formal method. Section 4 gives a browse of each level of the proposed Event-B specification process. Proof obligations are mentioned in Section 5. Finally, the conclusion and some research perspectives are given in Section 6.

II. Role-Based Access Control

In this section, an overview of the ANSI RBAC Standard (Institute, 2004; Institute, 2012) is given. This standard is defined by five sets: i) users (USERS); ii) roles (ROLES); iii) objects (OBJS); iv) actions or operations (OPS); v) sessions (SESSIONS).The RBAC concept is defined with permissions whom are granted to users through affected roles. Operations are applied on objects and sessions define the period of time when users can activate roles. An important aspect of RBAC is the definition of constraints on role activation. In term of these constraints, RBAC defines the notion of Separation of Duty (SoD). SoD relationships are used to apply conflict-of-interest management policies. Organizations employ these policies to prevent users from exceeding a reasonable level of authority for their positions. Two types of SoD are defined to avoid possible conflicts in the assignment of roles in RBAC: Static Separation of Duty (SSD) and Dynamic Separation of Duty (DSD).

- The SSD prevents the assignment of conflicted roles to the same user. Even in the hierarchy, a descendant role cannot be assigned to a user if its ascendant is in conflict of interest with another assigned role.
- The DSD prevents the assignment of conflicted roles to a user in the same time, during the allocation of a session. The solution of this problem can be resolved by assigning the conflicted roles to the user but in different sessions.

Figure 1 depicts the main RBAC entities and their associations [2]. In RBAC, user-to-role and permission-to-role assignments can be many-to-many. Thus, the same user can be assigned to many roles and a single role can be assigned to many users. Similarly, a single permission can be assigned to many roles and a single role can be assigned to many permissions.

TABLE I
The Event-B Development Structure

Context ctxt_a Extends ctxt_b Sets s Constants c Axioms A(s,c) Theorems T(s,c) End	Machine mach_a Refines mach_b Sees ctxt_a Variables v Invariants I(s,c,v) Theorems T(s,c,v) Variants V(s,c,v) Events evt= Any x Where G(s,c,v,x) Then v :| BA(s,c,v,x,v') End End

III. Event-B Method

Event-B is a formal method used to model and analyze systems. The key features of Event-B are in the use of:

- Set theory as a modeling notation,
- Refinement, to represent systems at different abstraction levels,
- Mathematical proofs, to verify the consistency between refinement levels.

An Event-B development model [18] is based on components of two kinds: Contexts and Machines. Contexts contain the static parts (axiomatization and theories) of the model, whereas the Machines implement the dynamic parts (states and transitions). Machines and contexts have various relations: a machine can be refined by another one, and a context can be extended by another one. Moreover, a machine can see one or several contexts. A Context is defined by the following clauses : CONTEXT, EXTENDS, SETS, CONSTANTS, AXIOMS and THEOREMS. Similarly to Contexts, Machines are defined by a set of clauses : MACHINE, REFINES, SEES, VARIABLES, INVARIANTS, THEOREMS and VARIANT.

The general structure of an Event-B development is illustrated in the Table I, where s denotes sets, c denotes constants and v denotes the declared variables of the machine. Axioms are denoted by $A(s,c)$ and theorems by $T(s,c)$, whereas invariants are denoted by $I(s,c,v)$ and local theorems by $T(s,c,v)$. For an event evt, its guards are denoted by $G(s,c,v,x)$ and its actions by the before-after predicate $BA(s,c,v,x,v')$.

The $EVENTS$ clause defines a list of events (transitions) that can occur in a given model. Each event is characterized by its guards and is described by a set of actions (substitutions). Each machine must contain an initialization event. The events occurring in an Event-B model affect the state described in clause. An event consists of the following clauses:

- Refines : declares a list of events refined by the described event.
- Any : lists a set of the event parameters.
- Where : expresses a set of guards for the event. An event can be fired when its guard turns to true. If several guards of events become true, only a single event is fired.
- Then : contains a set of actions of the event that are used to modify variables.

ISSN 2356-5608

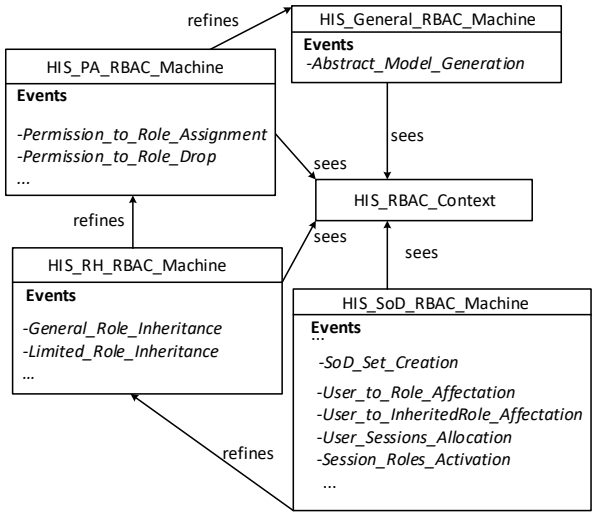

Fig. 2. The proposed Event-b specification

Event-B is based on a refinement methodology, it allows the system developer to start with an abstract model of the system considering its context, and gradually add details to the model. This process leads to a sequence of concrete models until the final implementation is reached. A number of proof obligations are generated through this process, which guarantees the correctness of the model as well as any desired invariants (properties) that the model should preserve.

IV. Event-B Specification of RBAC for Healthcare Information Systems

The proposed RBAC specification is detailed in this section. In the suggested approach, properties of the model are given in conjunction with the behavior specification. The model is developed in a way to link up between behavior and properties of RBAC components. Following a correct-by-design approach, validity and correctness of the RBAC properties definition are guaranteed through the specification process. In order to simplify the construction of the model, accesses are defined only by PERMISSIONS without splitting them on OPERATIONS and OBJECTS. The developed Event-B model contains one context that forms the static part of the specification, and four machines which form the dynamic part. Each machine expresses a level of the model specification. Figure 2 gives the structure of the proposed specification.

- $HIS\_RBAC\_Context$: The basic elements of RBAC are introduced in the static part of the specification. All the required static definitions to operate the dynamic part of the model are declared in this context (see Table II). In $HIS\_RBAC\_Context$, the declared working sets are: $HIS\_PERMISSIONS$,HIS_ROLES , HIS_USERS and $HIS\_SESSIONS$. The single condition on the definition of these sets is stated as an axiom: The sets must not be empty. The model is designed for a surgical clinic, where the medical staff is already organized into roles and the users achieve tasks (permissions) according to the roles granted to them. The process to be instantiated involves a Patient who has come to a surgical clinic to perform an operation. He possesses a file created by a Secretary, whom the Doctor can read or modify; the Patient must be operated by a Surgeon under anesthesia that the Anesthesiologist administered to him with the supervision of the ChiefDoctor. After this, the Nurse and the Doctor will take care of the postoperative follow-up of the Patient by checking and monitoring his condition but also treating his wounds. The process is divided into four sessions that users can allocate according to the involved roles.

TABLE II
The HIS RBAC Context

CONTEXT	HIS_RBAC _Context
SETS	HIS_USERS, HIS_ROLES, HIS_PERMISSIONS, HIS_SESSIONS
CONSTANTS	
	check,operate, treat, create_files, read_files, modify_files, supervise. Anesthetize, ChiefDoctor, Doctor, Nurse, Surgeon, Anesthesiologist, Patient, Secretary. user1, user2, user3, user4, user5 registration, diagnosis, surgery, treatment
AXIOMS	
axm1 :	partition(HIS_PERMISSIONS, {check}, {operate}, {treat}, {anesthetize}, {create_files}, {read_files}, {modify_files}, {supervise}.
axm2 :	partition(HIS_ROLES, {ChiefDoctor}, {Doctor}, {Surgeon}, {Anesthesiologist}, {Nurse}, {Patient}, {Secretary})
axm3 :	partition(HIS_USERS, {user1}, {user2}, {user3} , {user4}, {user5})
axm4 :	partition(HIS_SESSIONS, {registration}, {diagnosis}, {surgery}, {treatment})
Axm5 :	HIS_USERS $\neq \varnothing \wedge$ HIS_ROLES $\neq \varnothing \wedge$ HIS_PERMISSIONS $\neq \varnothing \wedge$ HIS_SESSIONS $\neq \varnothing$
END	

- $HIS\_General\_RBAC\_Machine$: The dynamic specification starts with a level that gives a global and high view of RBAC. It is expressed by the machine $HIS\_General\_RBAC\_Machine$ as depicted in the Figure III.
  In this level, details are not important; the RBAC structure has just to be in a brief and perceivable view, the considered properties are about the components typing and the definition of relations between them. The dynamic RBAC components are expressed by the variables: PA, RH, UA, US and RS. PA for permission to role assignment, RH for the role hierarchy definition, UA for the affectation of roles to users, US for the allocation of sessions by users and RS for the activation of roles during a session. These variables are manipulated by the event $Abstract\_Model\_Generation$ that generates

ISSN 2356-5608

TABLE III
THE HIS GENERAL RBAC MACHINE

HIS_General_RBAC_Machine
...
INVARIANTS
Inv1 : PA ⊆ HIS_PERMISSIONS × HIS_ROLES
Inv2 : RH ⊆ HIS_ROLES × HIS_ROLES
Inv3 : UA ⊆ HIS_USERS × HIS_ROLES
Inv4 : US ⊆ HIS_USERS × HIS_SESSIONS
Inv5 : RS ⊆ HIS_SESSIONS × HIS_ROLES
EVENTS
...
Abstract_Model_Generation
ANY
permissions, roles, users, sessions
WHERE
grd1: permissions ⊆ HIS_PERMISSIONS
grd2: roles ⊆ HIS_ROLES
grd3: users ⊆ HIS_USERS
grd4: sessions ⊆ HIS_SESSIONS
THEN
act1: PA :∈ ℙ(permissions×roles)
act2: RH :∈ ℙ(roles × roles)
act3: UA :∈ ℙ(users × roles)
act4: US :∈ ℙ(users × sessions)
act5: RS :∈ ℙ(sessions × roles)
End
End

TABLE IV
THE PERMISSION TO ROLES ASSIGNMENT MACHINE

HIS_PA_RBAC_Machine
...
INVARIANTS
...
Inv6 : PA1 ⊆ HIS_PERMISSIONS × HIS_ROLES
Inv7 : permissions1 ⊆ HIS_PERMISSIONS
Inv8 : roles1 ⊆ HIS_ROLES
EVENTS
...
Permissions_to_Roles_Assignment
ANY permission, role
WHERE
grd1: permission ∈ HIS_PERMISSIONS
grd2: role ∈ ROLES
THEN
act1: permissions1 := permissions1 ∪ { permission }
act2: roles1 := roles1 ∪ { role }
act3: PA1 := PA1 ∪ { permission ↦ role }
END

abstract views of the model. To preserve the abstraction, non-deterministic assignments are used. These affectations will gradually be determined in the refinement process.

- $HIS\_PA\_RBAC\_Machine$: This level provides more precision on the PA relation. As illustrated in the Figure IV, the variable $PA1$ concretizes PA by specifying roles and permissions that are involved in the relation. In order to preserve the typing properties of permissions and roles, witnesses are used in the refinement of the event $Abstract\_Model\_Generation$ as shown in the Figure V. The events where are handled the variables PA, $roles$ and $permissions$ ($Permissions\_to\_Roles\_Drop$, $Role\_Drop$,) are also declared in this machine.
- $HIS\_RH\_RBAC\_Machine$: This machine introduces the notion of hierarchy between roles. The event $General\_Role\_Inheritance$ allows a role who is hierarchically beyond another role to get all of his permissions. The principle guard of this event ($grd4$)states that both of the two roles must have been assigned to permissions. The concrete variable $RH1$ contains the inheritance couples as illustrated in the Figure VI.
  Besides the general role hierarchy defined in the RBAC Standard, there is the limited role hierarchy. The event $Limited\_Role\_Inheritance$ detailed in the Figure VII, states that for any different two roles $role1$, $role2$: if $role1$ inherits $role2$; there is no other role $role3$ that is inherited by $role1$ in the relation $RH1$.
  The event $Abstract\_Model\_Generation$ is concretized by refinement in each machine of the specification.
- $HIS\_SoD\_RBAC\_Machine$: The event $SoD\_Set\_Creation$ (Figure VIII) generates the set SOD that is composed of conflicted roles, sorted in couples (rs, n) .Where rs is the subset of conflicted roles and n is the number of roles that can be assigned.
- $UA\_RBAC\_Machine$: Users are introduced in this level, and are affected to roles with the event: $Users\_to\_Roles\_Affectation$. The variable UA is instantiated as $UA1$ after conditions have been defined

TABLE V
REFINEMENT OF THE EVENT ABSTRACT_MODEL_GENERATION

Abstract_Model_Generation
REFINES Abstract_Model_Generation
ANY users, sessions
WHERE
...
grd3: dom(PA1) ⊆ permissions1 ∧ ran(PA1)⊆roles1
WITH
permissions: permissions=permissions1
roles: roles=roles1
THEN
act1: PA :=PA1
...
End
...

TABLE VI
ROLE HIERARCHY MACHINE

HIS_RH_RBAC_Machine
...
EVENTS
...
General_Role_Inheritance
ANY role1, role2
WHERE
grd1: role1 ∈ HIS_ROLES ∧ role2 ∈ HIS_ROLES
grd2: role1 ∈ roles1 ∧ role2 ∈ roles1
grd3: role1 ≠ role2
grd4:(PA1 ▷ {role1} ≠ ∅) ∧ (PA1 ▷ {role2} ≠ ∅)
THEN
act1: RH1 := RH1 ∪ {role1 ↦ role2}
END
...

ISSN 2356-5608

TABLE VII
LIMITED ROLE INHERITANCE

```
...
Limited_Role_Inheritance
        ANY role1, role2
        WHERE
        grd1:role1 ∈ roles1 ∧ role2 ∈ roles1 ∧ role1≠role2
        grd2: {role1 ↦ role2} ⊆ RH1 ∧
        grd2: {role1 ↦ role2} ⊆ RH1 ∧
              ¬(∃ role3 · RH1▷{role1}={role3})
        THEN
        act1: RHL := RHL ∪ {role1}
END
...
```

TABLE VIII
HIS_SOD_RBAC_MACHINE

```
HIS_SoD_RBAC_Machine
...
INVARIANTS
              ...
              Inv10: SOD ⊆ ℙ(HIS_ROLES) × ℕ
              Inv11: users1 ⊆ HIS_USERS
...
EVENTS
...
SoD_Set_Creation
              ANY rs, n
              WHERE
              grd1: rs ⊆ HIS_ROLES ∧ n ∈ ℕ
              grd2: finite(rs) ∧ (n≥2 ∧ n ≤ card(rs))
              THEN
              act1: SOD := SOD ∪ {rs↦n}
END
...
```

when selecting the roles to assign. The guard $grd4$ of the Figure IX ensures that the roles that are (or must be) assigned to a user and those of the subset t do not exceed $(n-1)$. In other words, a user cannot be affected to more than $(n-1)$ conflicted roles of rs.

The event $User\_to\_InheritedRole\_Affectation$, depicted in the Figure IX, allows the assignment of a descendant role to a user if its' ascendant is assigned to this same user. Considering that the separation of duty is categorized according to abstraction levels in the proposed model, the $grd4$ shows that a user cannot be assigned to more than $(n-1)$ role of rs, the same way as in the preceding event. The difference is that the role from rs has an inheritance link in $RH1$. Sessions are allocated with the event $User\_Sessions\_Allocation$ depicted by the Figure X. The assignment of sessions to a user $(user, session)$ is contained in the concrete variable $US1$.

V. PROOF OBLIGATIONS OF THE MODEL

The entire Event-B model presented above has been developed within the Rodin platform. This latter generates automatically Proof Obligations in the form of sequences [18]. The automatic Prover of RODIN can discharge automatically many of the POs, the remainder of non-discharged POs can be tackled by the interactive Prover. The developed model led

TABLE IX
EVENTS FOR THE STATIC SEPARATION OF DUTY

```
...
User_to_Role_Affectation
ANY user, role, rs, n, t
WHERE
grd1: user ∈ HIS_USERS ∧ rs ∈ dom(SOD)
grd2: finite (rs) ∧ t⊆ rs ∧ n≥2 ∧ n ≤ card(rs)
grd3: finite(t) ∧ card(t)≥n ∧ role ∈ t
grd4: card(UA1[{ user }] ∩ t)≤ n−1
THEN
act1: users1 := users1 ∪ {user}
act2: UA1 := UA1 ∪ {user↦role}
END

User_to_InheritedRole_Affectation
ANY user, role, role1, rs, n, t
WHERE
grd1: user ∈ HIS_USERS ∧ user ∈ users1 ∧ n ∈ ℕ
grd2: finite(rs) ∧ rs ∈ dom(SOD) ∧finite(t) ∧
      t ⊆ rs ∧(n≥2 ∧ n≤card(rs) ∧ card(t)≥n)
grd3: user↦role⊆ UA1 ∧ {role↦role1}⊆ RH1
grd4: card(UA1[{user}] ∩ t) ≤ n−1
THEN
act1: UA1 := UA1 ∪ {user↦role1}
END
```

TABLE X
USER_SESSIONS_ALLOCATION EVENT

```
...
User_Sessions_Allocation
        ANY user, session
        WHERE
        grd1: user ∈ HIS_USERS ∧ session ∈ HIS_SESSIONS
        grd2: user ∈ users1
        THEN
        act1: sessions1:= sessions1 ∪ {session}
        act2: US1 := US1 ∪ {user↦session}
END
...
```

to 32 proof obligations. 31 were proved automatically and one needed few interactive proof steps. Table XI gives the details of these results. As can be seen, the adopted specification approach engendered a few number of POs since the properties were expressed in the events' guards. Accordingly, several POs were automatically guaranteed by construction.

VI. CONCLUSION

In this work, an approach is presented to design an RBAC model for healthcare systems. Due to the limitation of approaches that are based on a-posteriori checking process and in order to deal with the validation of large-scale systems, a correct-by-design approach is proposed. Accordingly, we propose an Event-B specification of the RBAC model for a

TABLE XI
STATISTIC OF PROOFS

Model components	PO	Automatic proof	Interactive proof
HIS_General_RBAC_Machine	5	5	0
HIS_PA_RBAC_Machine	7	7	0
HIS_RH_RBAC_Machine	2	2	0
HIS_SoD_RBAC_Machine	18	17	1

ISSN 2356-5608

healthcare system where all the properties are validated and the correct behavior is proved. The main features of the solution are:

- The model is designed in stepwise manner, with refinements and proving-based specification.
- The approach generates specifications with different abstraction views which simplify the observation and analysis of the specified model.

As future work, the aim would be to:

- Extend the approach to allow model reconfiguration;
- Apply the approach to other access control paradigms [26].

REFERENCES

[1] A. N. S. Institute, *Role Based Access Control, INCITS 359-2004*, ANSI, 1430 Broadway, New York, NY 10018, USA, 2004.

[2] ——, *Role Based Access Control, INCITS 359-2012*, ANSI, 1430 Broadway, New York, NY 10018, USA, 2012.

[3] A. C. OConnor and R. J. Loomis, "2010 economic analysis of role-based access control," *NIST, Gaithersburg, MD*, vol. 20899, 2010.

[4] E. Ferrari, "Guest editorial: Special issue on access control models and technologies," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 4, pp. 349–350, 2005.

[5] T. Ahmed and A. R. Tripathi, "Static verification of security requirements in role based cscw systems," in *Proceedings of the eighth ACM symposium on Access control models and technologies*. ACM, 2003, pp. 196–203.

[6] M. Drouineaud, M. Bortin, P. Torrini, and K. Sohr, "A first step towards formal verification of security policy properties for rbac," in *Quality Software, 2004. QSIC 2004. Proceedings. Fourth International Conference on*. IEEE, 2004, pp. 60–67.

[7] G. Hughes and T. Bultan, "Automated verification of access control policies using a sat solver," *International journal on software tools for technology transfer*, vol. 10, no. 6, pp. 503–520, 2008.

[8] L. Kahloul, K. Djouani, W. Tfaili, A. Chaoui, and Y. Amirat, "Modeling and verification of rbac security policies using colored petri nets and cpn-tool," in *International Conference on Networked Digital Technologies*. Springer, 2010, pp. 604–618.

[9] P. Konopacki, M. Frappier, and R. Laleau, "Expressing access control policies with an event-based approach," in *International Conference on Advanced Information Systems Engineering*. Springer, 2011, pp. 607–621.

[10] M. Koch, L. V. Mancini, and F. Parisi-Presicce, "A graph-based formalism for rbac," *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, no. 3, pp. 332–365, 2002.

[11] B. Shafiq, A. Masood, J. Joshi, and A. Ghafoor, "A role-based access control policy verification framework for real-time systems," in *Object-Oriented Real-Time Dependable Systems, 2005. WORDS 2005. 10th IEEE International Workshop on*. IEEE, 2005, pp. 13–20.

[12] N. Al-Hadhrami, B. Aziz, and L. ben Othmane, "An incremental b-model for rbac-controlled electronic marking system," *International Journal of Secure Software Engineering (IJSSE)*, vol. 7, no. 2, pp. 37–64, 2016.

[13] C. Yuan, Y. He, J. He, and Z. Zhou, "A verifiable formal specification for rbac model with constraints of separation of duty," in *International Conference on Information Security and Cryptology*. Springer, 2006, pp. 196–210.

[14] H. Hu and G. Ahn, "Enabling verification and conformance testing for access control model," in *Proceedings of the 13th ACM symposium on Access control models and technologies*. ACM, 2008, pp. 195–204.

[15] H. Ferrier-Belhaouari, P. Konopacki, R. Laleau, and M. Frappier, "A design by contract approach to verify access control policies," in *Engineering of Complex Computer Systems (ICECCS), 2012 17th International Conference on*. IEEE, 2012, pp. 263–272.

[16] N. Li, J.-W. Byun, and E. Bertino, "A critique of the ansi standard on role-based access control," *IEEE Security & Privacy*, vol. 5, no. 6, 2007.

[17] N. Huynh, M. Frappier, A. Mammar, R. Laleau, and J. Desharnais, "A formal validation of the rbac ansi 2012 standard using b," *Science of Computer Programming*, vol. 131, pp. 76–93, 2016.

[18] J. Abrial, "Modeling in event-b: System and software development," 2010.

[19] T. S. Hoang, D. Basin, and J.-R. Abrial, "Specifying access control in event-b," *Technical report*, vol. 624, 2009.

[20] M. Leuschel and M. Butler, "Prob: A model checker for b," in *International Symposium of Formal Methods Europe*. Springer, 2003, pp. 855–874.

[21] S. Schneider, *The B-method: An introduction*. Palgrave, 2001.

[22] Z. Farah, Y. Ait-Ameur, M. Ouederni, and K. Tari, "A correct-by-construction model for asynchronously communicating systems," *International Journal on Software Tools for Technology Transfer*, vol. 19, no. 4, pp. 465–485, 2017.

[23] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, 2015.

[24] H. Gadouche, F. Zoubeyr, and A. Tari, "A correct-by-construction model for attribute-based access control," in *MEDI*, 2018.

[25] J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin, "Rodin: an open toolset for modelling and reasoning in event-b," *International journal on software tools for technology transfer*, vol. 12, no. 6, pp. 447–466, 2010.

[26] A. A. E. Kalam, S. Benferhat, A. Miège, R. E. Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and G. Trouessin, "Organization based access control," in *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, ser. POLICY '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 120–. [Online]. Available: http://dl.acm.org/citation.cfm?id=826036.826869

ISSN 2356-5608