Incremental Development of RBAC-controlled E-marking System Using the B Method

Abstract— **Due to the significance of using formal methods in software development, it is helpful to be used as well to verify the correctness of specifications of some security policies, such as RBAC model, to be an important step before the actual implementation to systems. Using formal methods to re-formulate security's specification in a mathematical way would make the policy more reliable when it comes to the application of the policy to an appropriate system. This paper shows the mechanism of formalizing RBAC policies of an Electronic Marking System (EMS) using B specifications. It also illustrates some approaches to verify the consistency of the RBAC specification, such as model checking and proof obligations.**

Keywords- Role Based Access Control (RBAC); formal specifications; RBAC constraints; separation of duties; role hierarchy; cardinality constraints; model checking; proof obligations

I. INTRODUCTION

Security is now becoming a very important aspect when it comes to the development of modern systems. Many critical systems, especially those which handle commercial transactions, enforce different security policies to their systems in order to ensure that existing data and information flow are enough secure to preserve confidentiality. If there is any defect in applying any security policy to a particular system, as if specifying ambiguous properties or defining inconsistent model, then the system might lack reliability. Hence, formal methods are widely used to verify the correctness and consistency of most of security models, including RBAC policies, as they rely on *mathematical logic* and *set theory* [3].

Role Based Access Control (RBAC) model is regarded as a successful alternative to discretionary and mandatory access control models. Efficiently, RBAC can be more suitable for commercial systems, since it depends essentially on assigning different users to particular roles [4]. However, RBAC policy is not restricted to business systems only, but also to any critical system that needs to determine whether a given subject (user) is allowed to access a certain object (resource).

This paper discusses a study case: Electronic Marking System (EMS) in Oman Secondary Schools, where RBAC security policy needs to be applied to. EMS aims to provide a consolidated environment, where each member within a school has the right to access the system within the powers (authorizations) given to him/her. For example, teachers can access the system for the purpose of adding, editing or deleting marks. Whereas, students have the authorization to submit reports and view their grades.

II. RBAC SECURITY POLICY

Role Based Access Control (RBAC) is one of the most efficient models of access control policies [1]. Began in 1970s with multi-user and multi-application, and has rapidly evolved in the last three decades as a technology for applying a high level security in large-scale systems. The pivotal idea behind RBAC model is that permissions are associated with roles, and users are administratively assigned to proper roles [11]. This mechanism ensures that only authorized users can perform some functions on some data/resources [4]. Generally, RBAC model consists of some basic elements, such as Users, Roles, Permissions, User Assignment (UA), Permissions Assignment (PA) and Constraints (see Figure 1):

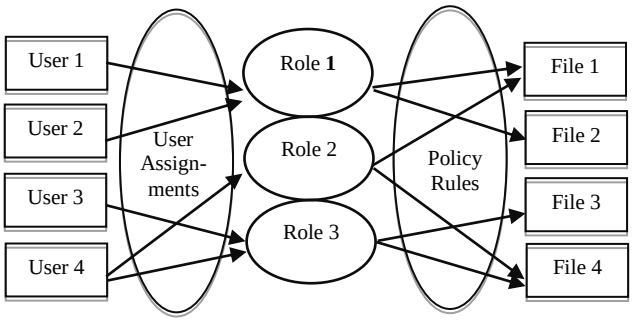

Figure 1: The concept of RBAC security policy

It is clear that users are not mapped directly into permissions of accessing some resources, but to specific roles which have to be previously assigned to those permissions [9]. For example, in medical systems, nurses are authorized to access specific resources (e.g. files) which certainly differ from those in which physicians are allowed to. Therefore, each role in the system (i.e. nurse and physician) is being mapped into the corresponding permissions. If Alice, for instance, is a nurse, then she has to be assigned to the role *nurse* [3].

In RBAC model, [7] states that there are some relevant terms and concepts, which define the different model's elements (see Figure 2). However, not all of them are necessary to represent an RBAC security policy, where there are some models that describe only certain needs for a particular system.

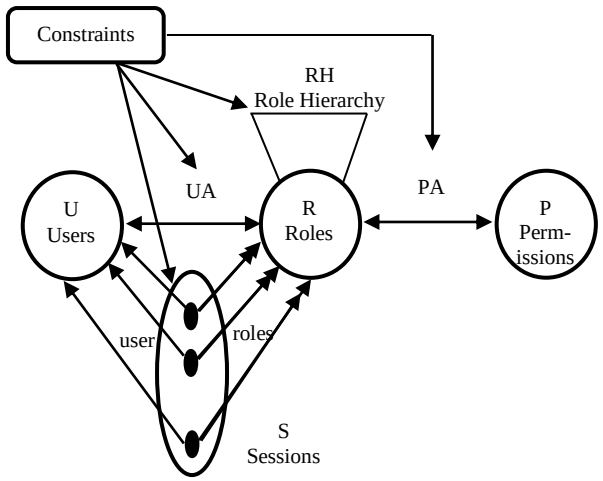

Figure 2: Elements of RBAC security policy

- Users (U): persons who interact with a system.
- Roles (R): prescribed behaviours, which describe particular positions or functions within an organization.
- Permissions (P): descriptions of the type of interactions that a user can have with an object.
- Object: a passive entity that has some information, and can receive new information.
- User Assignment (UA): a many-to-many relationship between users and roles.
- Permission Assignment (PA): a many-to-many relationship between permissions and roles.
- User Assignment (UA): a many-to-many relationship between users and roles.
- Permission Assignment (PA): a many-to-many relationship between permissions and roles.
- Session (S): a mapping between a user and any activated role/s that the user is assigned to.
- Role Hierarchy (RH): a partial order relationship on roles.
- Constraints: restrictions on any of the above relationships/assignments.

III. RBAC MODELS

There is a family of four conceptual models for RBAC security policy, which define the various dimensions of it [6]. $RBAC_0$ is considered as the basic model, which has the minimum requirements for a system. Whereas, $RBAC_1$, $RBAC_2$ and $RBAC_3$ are more advanced models. However, the latter models are based on the basic one '$RBAC_0$', as some elements are common for all types of models [1].

$RBAC_0$ contains some basic elements of RBAC policies, where cannot define any RBAC policy without specifying those elements. Usually, components of $RBAC_0$ are Users (U), Roles (R), Permissions (P), Sessions (S) as well as relationships between them.

In addition to the components of $RBAC_0$, $RBAC_1$ model supports the concept of Role Hierarchy (RH). Hierarchical of roles describe the gradualism of authorizations and responsibilities in an organization. In other words, RH is about defining what are called senior roles (more powerful roles) and junior roles (less powerful roles), where senior roles may inherit permissions that are assigned to junior roles.

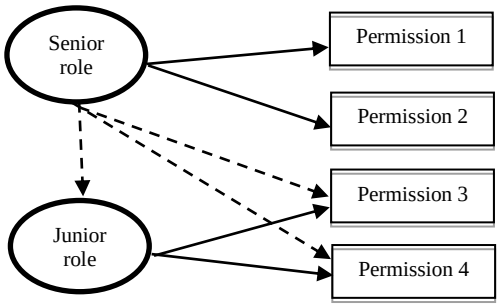

Figure 3: Role Hierarchy (RH)

In order to provide a powerful protection to the RBAC-based systems, $RBAC_3$ (constraints model) is defined to be the main drive of RBAC security policies. That is $RBAC_3$ uses a mechanism to force restrictions and form a high level security policy. Constraints can be applied to any relationship/assignment that has been defined in the two previous models, such as UA, PA, Sessions and Role Hierarchy (RH) (see Figure 4). There are many types of applicable constraints, in particular those which deal with exclusiveness of roles (Separation of Duties) and cardinality constraints.

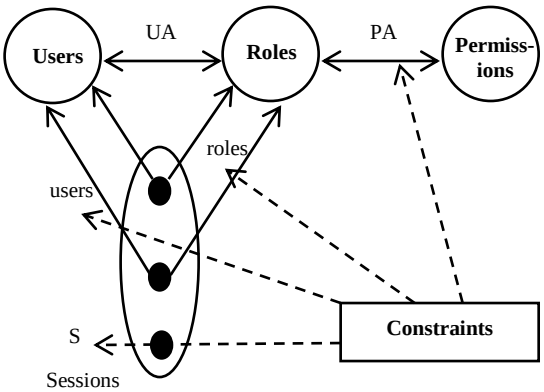

Figure 4: RBAC Model Constraints

Mutual exclusiveness of roles is one of the most common constraints for an RBAC model. Usually, it is defined as Separation of Duties (SoD), where the same user cannot be assigned to two conflicted roles at the same time [4]. For example, for two roles *R1* and *R2* belonging to mutually exclusive sets of roles, then if a user *U* is

assigned to *R1*, this implies that *U* cannot be assigned to *R2*, and vice versa. This is expressed formally as:

If: (U x R1) ∈ UA Then (U x R2) ∉ UA ; and
If: (U x R2) ∈ UA Then (U x R1) ∉ UA

Cardinality constraints define the maximum number of users that can be assigned to a particular role. Likewise, cardinality constraints can be used to specify the number of roles that can be related to a particular user. In addition to that, they can be applied to Permission Assignment (PA) for the purpose of restricting the number of permissions that a particular role should be assigned to [7].

IV. THE EMS SYSTEM AND IT'S ENTITIES

The Electronic Marking System (EMS) allows students, teachers, headmasters and guardians to access students' marks of their units. Each user is able to benefit from the system according to the powers/authorizations given to him/her. For example, students can electronically submit papers (reports) for a particular unit to be marked by the unit's teacher. Teachers, on the other hand, have the right to establish, edit and delete marks of their students. In addition, headmasters and guardians can also access the system for the purpose of viewing marks and signing final reports.

Generally, the EMS system can be accessed by different users: the system admin (who controls and manages the different entities and resources of the system), students, teachers, headmasters and guardians. Hence, users can be classified into three main entities (tables): staff, students, and guardians, where staff entity represents teachers and headmasters as well as the system admin.

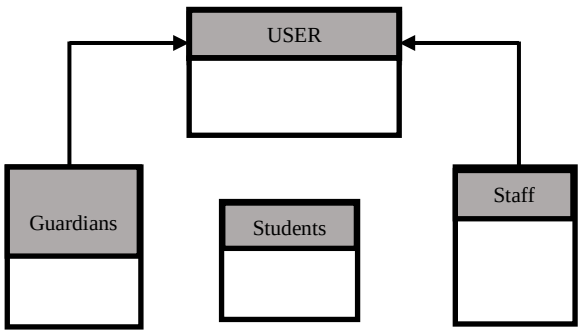

Figure 5: Entities of the EMS's users

The other entities of the system would be as follows:

- UNIT: to represent all units taught at secondary schools.
- SUBMISSION: to store submitted reports by students.
- FINAL_REPORT: which contains final reports for all students.
- MARK: its contents are natural numbers, which represent students' marks.
- ROLES: this entity is set up for the purpose of applying RBAC model. Hence, it represents the different roles in the EMS system (i.e. admin, teacher, student, etc.).

There are also relationships between those entities, which can be briefly stated as:

- studies: represents a relation between a student and a unit; where each student studies more than one unit; and that unit can be registered for many students.
- taughtBy: a relation between a unit and a teacher; where each unit must be taught by at least one teacher; and that teacher can be registered to more than one unit..
- hisGuardian: a relation between a student and a guardian. A student can have more than one guardian in order to facilitate the process of following up the student's level. In addition, the guardian can be registered for more than one student.
- submits: a relation between a student and a submission; where any student can submit any number of reports (depending on the unit and the academic level that a student is studying at); however, any report cannot represent a submission for more than one student.
- reportFor: a relation between a submitted report and a unit. Therefore, all submitted reports are for particular units, and each unit may have more than one report.
- submissionMark: a relation between a submitted report and a mark (which is a natural number). However, cannot find any report mapped into two different marks (axiomatically!), but we can find that a given mark represents the grade of more than one report.
- InClassMark: a relation between a unit and a mark (i.e. natural number) to specify a student's mark for his/her participations and activities inside the class during the year.
- examMark: a relation between a unit and a mark to specify the student's mark/s (depending on how many exams for that unit).
- studentReport: a relation between a student and a final report; where each student must have a final report for each unit, which shows his/her final mark. However, cannot find two final reports are related to the same student.

Therefore, the final mark of each student for a particular unit is being calculated as follows:

Final mark = total of submissions marks + total of exams' marks + in-class mark

- hasTheRole: maps a user (i.e. a student, a teacher, etc.) into a set of roles (e.g. student, teacher, headmaster, student guardian); where each user would be linked to only one role (unless there were some circumstances made otherwise, e.g. in case of absence of the school manager, any teacher can be assigned to play the role of headmaster; and this assignment falls under the responsibility of the system's admin).

V. Application of RBAC to the EMS

In order to apply an RBAC security policy to the EMS system, we need to define the different components of RBAC models (i.e. $RBAC_0$, $RBAC_1$ and $RBAC_2$). For $RBAC_0$, it is essential to specify the sets of *USERS*, *ROLES*, and *PERMISSIONS*.

A. $RBAC_0$- Basic model:

Since the system has three main types of users (i.e. staff, students, students'guardains), each user represents an element in the USER set (U). Regarding ROLES (R), there are six basic roles, which describe the main functionality for users in the Electronic Marking System (EMS). These roles would be stored into the ROLES entity of the database. Therefore, contents of this entity can be defined as follows:

ROLES = {admin, student, teacher, headteacher, headmaster, student_guardian}

Permissions (P) are pairs of (*operations*, *objects*) in which users are allowed to perform.. For example, a student wants to store a report (*operation*) into SUBMISSION table (*object*) in order for the report to be considered "submitted". This process means that the student is permitted to *add submission*. The EMS system allows each type of users to perform only specific permissions.

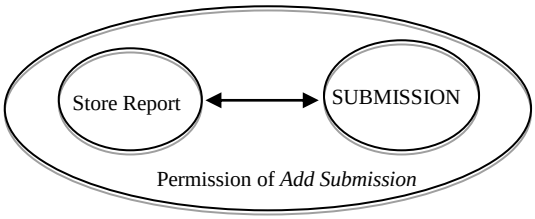

Figure 6 "*Add Submission*" permission for students

B. User Assignment (UA) and Permission Assignment (PA):

User Assignment (UA) is a many-to-many relationship between the system's users *U* and the roles *R*. For example, students take the role *student*, while teachers are assigned to role *teacher*. In some cases, it is possible to assign a user to two different roles if a given situation requires that. For instance, in case of absence of a headmaster, any teacher can be authorized to perform the headmaster permissions, and that would be done through giving the role *headmaster* to the teacher, in addition to his/her own role.

Permission Assignment (PA) is also a many-to-many relationships, in which roles are mapped into permissions (functions). There are specific functions for each role, where can only be performed by user(s) who is/are assigned to those roles. The following diagram shows an example of Admin assignment:

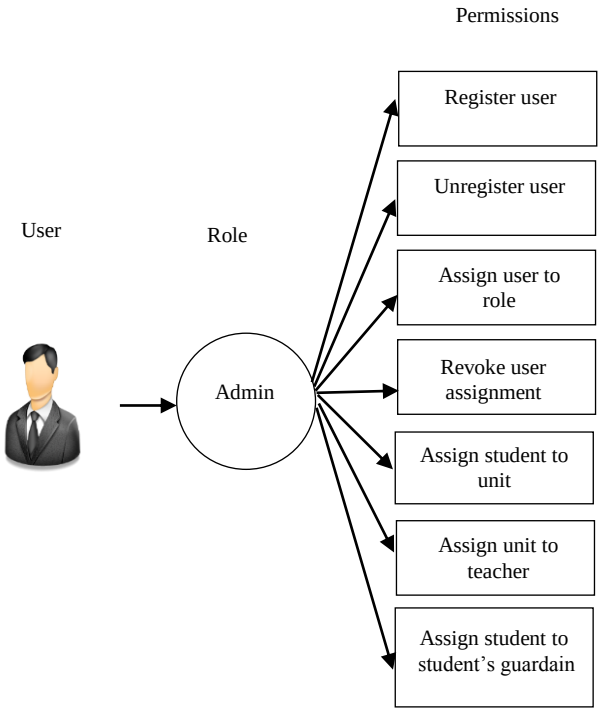

Figure 7: "Admin" role in the EMS System

C. $RBAC_1$- Role Hierarchy (RH):

In the EMS system, there is only one instance of role hierarchy. Head teachers are, in fact, teachers, although they have special tasks for organizing and monitoring roles assigned to teachers, such as reviewing records. When a head teacher is assigned to role *headteacher*, this means that the head teacher is allowed to perform functions of the *headteacher* role, and inherit permissions from the role *teacher*.

D. $RBAC_2$- Constraints:

For the EMS system, there are two types of constraints that can be defined: cardinality constraints and mutual exclusiveness of roles. Usually, cardinality constraints represent the number of users who should be assigned to a particular role. In our system, it doesn't matter how many students to be assigned to *student* role, or how many teachers to be mapped into *teacher* role, as that depends on

the number of students and teachers inside a certain school. However, there must be only one user who should take the role *headmaster*. Thus, the cardinality constraint for role *headmaster* is expressed as:

UA (User Assignment) ⊆ U (User) x R (Role); where cardinality (U(school manager) x (R(headmaster)) = 1

Mutual exclusiveness of roles (Separation of Duties) define the conflicted sets of roles, which must not be related to the same user/s. For example, if there are two roles *R1* and *R2* belonging to two conflicted sets of roles *S1* and *S2*, respectively, then: If a user *U* has been assigned to a role in *S1*, this implies that *U* is not assigned to any role in *S2*, and vice versa.

In the Electronic Marking System (EMS), there are two conflicted classes (sets) of roles:

SoD1 = {teacher, headteacher, headmaster}, and SoD2 = {student, student_guardian}

VI. Modeling the RBAC Policy Using the B Specification

Analyzing RBAC models using a formal method, such as the B method or Event-B may assist system developers to obtain a correct and consistent specification. By knowing RBAC constraints, formal specifications can play a significant role to examine security policies, and determine whether a particular system is secure at every point of time.

The B method is one of the abstract formal languages that uses a special language named "Abstract Machine Notation (AMN)". The idea is to use mathematical notations (relying on *mathematical logic* and *set theory*) to formalize specifications for the purpose of formally verifying correctness and consistency of a specification.

A. Modeling the Basic RBAC Components

To formalize the basic RBAC elements, it is essential to define the sets of *USER* and *ROLES* as well as *UA* and *PA* assignments. In the EMS system, *USER* set contains subsets of students, guardians and staff. The latter itself contains subsets of teachers and headmasters. Therefore, these subsets must be declared as *variables* in the AMN:

staff, students, studentGuarian, teachers, manager

Now, we can declare, within INVARIANT clause, that each of those variables is a subset of *USER* set, and some are also subsets of *staff* subset, provided that no element of each subset can be found into two different subsets. For example, if a student has been registered into *students* subset, then that student cannot be found as a member of *staff* subset (i.e. *students* ∩ *staff* = { }). This can be expressed using the B specification as follows:

$students \subseteq USER \ \wedge$
$staff \subseteq USER \ \wedge$
$students \cap staff = \{\} \ \wedge$
$studentGuardian \subseteq USER \ \wedge$
$studentGuardian \cap students = \{\} \ \wedge$
$studentGuardian \cap staff = \{\} \ \wedge$
$manager \subseteq staff \ \wedge$
$teachers \subseteq staff \ \wedge$
$manager \cap teachers = \{\}$

To model the User Assignment (UA), which assigns each user to a particular role, we need to define a relation between *USER* set and *ROLES* set. This relation is, in fact, a set of elements, where each element is a pair of user and role. For example, if a user *u* has been assigned to a role *r*, then there is an element for the relation can be defined as: $\{(u, r)\}$.

Since elements of the relation between *USER* and *ROLES* are subject to change (i.e. assigning new users to roles, or revoking existing assignments), we represent this relation as a variable, and we give it the name: *hasTheRole*. The declaration of that within the INVARIANT clause would be as:

$hasTheRole \in USER \leftrightarrow ROLES$

To model the Permissions Assignments (PA), we do not need to define a new set for permissions, or a direct relation (as in the UA assignment) between *ROLES* and those permissions. Since any permission is actually a pair of operation and object, we will be specifying (within preconditions of each operation (function)) in the system, that this operation is only performed by users who have been given a particular role.

For example, users who can add a submission are only students. Therefore, we say, as a precondition of the operation, that "*who has the role of 'student', can perform this operation*". This is expressed by the B specification as in Figure 8:

addSubmission (s, report, u) ≙
PRE
hasTheRole[{s}] = {student} ∧
$report \in SUBMISSION \wedge$
$report \notin reports \ \wedge$
$u \in studies[\{s\}] \ \wedge$
$card(submit[\{s\}] \cap reportFor^{-1}[\{u\}] < maxSubmissions$
THEN
$report := reports \cup \{report\} \ ||$
$submits := submits \cup \{s \mapsto report\} \ ||$
$reportFor := reportFor \cup \{report \mapsto u\}$
END;

Figure 8: Assigning "*add a submission*" permission to "*student*" role

B. Modeling the RBAC Constraints:

We begin with mutual exclusiveness of roles (Separation of Duties). As has been mentioned earlier, there are two conflicted sets of roles in the EMS system. Since roles are previously known and defined for the system, the two conflicted sets of roles should be initialized within the INITIALIZATION clause as:

$$sod_1 = \{teacher, headteacher, headmaster\},$$
$$sod_2 = \{student, student\_guardian\}$$

Each of sod_1 and sod_2 is a subset of *ROLES* set, and the intersection between them is equal to an empty set. Therefore, this must be declared within *INVARIANT* clause:

$$sod1 \subseteq ROLES \ \wedge$$
$$sod2 \subseteq ROLES \ \wedge$$
$$sod1 \cap sod2 = \{\ \}$$

Now, we can formulate the condition of Separation of Duties (SoD) using the B specification, as follows:

$$\forall(u, role1, role2).\ (u \in USER \ \wedge \ role1 \in sod1 \ \wedge \ role2 \in sod2 \ \wedge$$
$$role1 \in hasTheRole[\{u\}] \Rightarrow not\ (role2 \in hasTheRole[\{u\}]\ ) \ \wedge$$

$$\forall(u, role1, role2).\ (u \in USER \ \wedge \ role1 \in sod1 \ \wedge \ role2 \in sod2 \ \wedge$$
$$role2 \in hasTheRole[\{u\}] \Rightarrow not\ (role1 \in hasTheRole[\{u\}]\ )$$

The above two predicates mean, respectively, that for every user *u* and two given roles *role1* and *role2*, where *u* ∈ *USER* and *role1* ∈ *sod1* and *role2* ∈ *sod2* and that *role1* belongs to the set of roles given to the user *u*, then the *role2* must **not** belong to the roles of the user *u*. Likewise, for every user *u* and two given roles *role1* and *role2*, where *u* ∈ *USER* and *role1* ∈ *sod1* and *role2* ∈ *sod2* and that *role2* belongs to the set of roles given to the user *u*, then the *role1* must **not** belong to the roles of the user *u*

The second type of constraints is cardinality constraints. In the EMS system, the number of users who will be assigned to *headmaster* role must not be greater than 1. Hence, the cardinality of users who have the role *headmaster* must be less than or equal to 1. This is expressed as:

$$\forall(headmaster).\ (headmaster \in sod1 \Rightarrow card(hasTheRole^{-1}[\{headmaster\}] \leq 1\ )$$

C. Modeling the Role Hierarchy (RH):

Head teachers have their own permissions, such as reviewing records. However, they still teachers, and, therefore, can perform operations that are given to teachers. We need to formulate a predicate that satisfies the concept of inheritance, where head teachers can inherit permissions from *teacher* role. In the B specification, this is expressed as:

$$\forall(u, headteacher).\ (headteacher \in sod1 \ \wedge \ u \mapsto headteacher \in hasTheRole \Rightarrow headteacher \in hasTheRole[\{u\}] \ \wedge \ teacher \in hasTheRole[\{u\}]\ )$$

Which means that for every user *u* and *headteacher* role, where *headteacher* ∈ *sod1* and that user has been assigned to the role *headteacher*, then the user *u* would become have the role *teacher* in addition to the role *headteacher*.

VII. Formal Verification

Perhaps the most important aspect in using formal methods in software development is that specifications can be analyzed and verified using mathematical techniques and tools. There are two approaches for formal verification: model checking and theorem proving (e.g. proof obligations).

A. Model Checking (the proB tool):

Model checking is an automated approach that used to verify correctness and consistency of a model. The *proB* tool is one of the B machines-supported tools that test automatically the consistency of the B specification. The main idea of model checking is that all machine's nodes (states) are visited to examine whether there exist any potential bugs [5].

The *proB* tool showed that our RBAC specification is correct regarding the system states (nodes). There was no violation with the system properties and invariant when it comes to the performance of operations.

B. Theorem Proving (Proof Obligations):

The main purpose of theorem proving is to construct a *mathematical proof* for a mathematical statement to be true [2]. Since formal proofs do not use natural languages, they are expressed in a symbolic language, usually called a proof language. For abstract machines to be correct and consistent, there are many proof obligations that need to be proven. Theses proof obligations are for the four main clauses of an abstract machine: *PROPERTIES*, *INVARIANT*, *INITIALISATION* and *OPERATIONS*, where there is a separate proof obligation for each operation in the machine [5].

Since our RBAC model doesn't have properties clause, proof obligations would be for the clauses of *INVARIANT*, *INITIALISATION* and *OPERATIONS*. Proof obligation for invariant clause is expressed as:

$$Prp \Rightarrow \exists v.\ I$$

Which means that under the assumption that *Prp* (PROPERTIES clause, if found) is true, then: the

invariant *I* is satisfied by at least one of the machine's variables *v*

To discharge this obligation, we need to show that there exist values for:

- *sod1* and *sod2*, which are subsets of *ROLES*; where the intersection between them is an empty set.
- *hasTheRole*, which is a relation between *USER* and *ROLES*; where:
  - for any user and two conflicted roles, the concept of SoD is applied;
  - for any user has the role *headteacher*, the concept of Role Hierarchy is applied; and
  - cardinality of users who take the role *headmaster* is less than or equal to 1.
- *staff*, *teachers*, *manager*, *students* and *studentGuardian*, which are subsets of *USER*; where:
  - *teachers* and *manager* are subsets of *staff*; and
  - the intersection between any two subsets is an empty set.
- *hisGuardain*, which is a partial surjective function between *students* and *studentGuardain*.

Perhaps the easiest way to demonstrate that is to apply the current values of variables in the *INITIALISATION* clause (i.e. all variables are equal to { }, except the sets of *sod1* and *sod2*).

The proof obligation for INITIALIZATION takes the following predicate:

$$Prp \Rightarrow [Init]\ I$$

Which means that under the assumption that *Prp* (PROPERTIES clause, if found) is true, then: the initialisation clause *Init* must establish the invariant *I*. To demonstrate this obligation, we apply the current values in the *INITIALISATION* clause to their corresponding variables in the *INVARIANT* clause.

For each operation in the machine, there must be a separate proof obligation. Hence, there will be proof obligations for *assignUserToRole* and *revokeUserAssignment* operations. However, we will discuss only the proof obligation for *assignUserToRole* operation. This takes the following predicate:

$$I \wedge P \Rightarrow [S]\ I$$

Which means that under the assumption that invariant *I* is true and the preconditions *P* of the operation is true as well, then: the statement of the operation *S* (i.e. the operation's body) must preserve the invariant *I*.

VIII. Conclusion

RBAC security policy is considered as one of the most effective security models to different critical systems, such as business systems. Formal specification is now widely used to verify security constraints/properties of RBAC models in different ways and several languages. This paper discussed the use of the B method to develop a consistent specification for the RBAC security policy of the Electronic Marking System (EMS). There was the use of two approaches for formal verification: model checking and proof obligations. Model checking was represented by the application of *proB* tool, while proof obligations constructed a mathematical statement/proof for the RBAC model to be true. Both mechanisms proved that our RBAC specification is correct and consistent regarding the implementation of the system operations.

References

[1] Ahn, G. J., & Hu, H. (2007, June). Towards realizing a formal RBAC model in real systems. In *Proceedings of the 12th ACM symposium on Access control models and technologies* (pp. 215-224). ACM

[2] Boulanger, J. (2012). *Industrial Use of Formal Methods: Formal Verification*. London: ISTE; Hoboken

[3] Drouineaud , M., Bortin, M., Torrini, P., & Sohr , K. (2004). A first step towards formal verification of security policy properties for RBAC. *Proceedings of the Fourth International Conference on Quality Software (QSIC'04)*, Retrieved from http://www.cs.le.ac.uk/people/pt95/IEEEXplore.pdf

[4] Ferraiolo, D. F., & Kuhn, D. R. (2009). Role-based access controls. *arXiv preprint arXiv:0903.2171*. Retrieved from http://arxiv.org/ftp/arxiv/papers/0903/0903.2171.pdf

[5] Mentré, D., Marché, C., Filliâtre, J. C., & Asuka, M. (2012). Discharging proof obligations from Atelier B using multiple automated provers. In *Abstract State Machines, Alloy, B, VDM, and Z* (pp. 238-251). Springer Berlin Heidelberg.

[6] Pham, T. H., Truong, N. T., & Nguyen, V. H. (2009, October). Analyzing RBAC security policy of implementation using AST. In *Knowledge and Systems Engineering, 2009. KSE'09. International Conference on* (pp. 215-219). IEEE.

[7] Sohr, K., Drouineaud, M., Ahn, G. J., & Gogolla, M. (2008). Analyzing and managing role-based access control policies. *Knowledge and Data Engineering, IEEE Transactions on, 20(7)*, 924-939. Retrieved from http://www.db.informatik.uni-bremen.de/publications/Sohr_2008_TKDE.pdf

[8] Strembeck, M., & Neumann, G. (2004, August). An Integrated Approach to Engineer and Enforce Context Constraints in RBAC Environments. *This paper appeared in the Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003), 3*, 392–427. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.7854&rep=rep1&type=pdf

[9] Yu, S., & Brewster , J. (2012, April). Formal Specification and Implementation of RBAC Model with SOD. *JOURNAL OF SOFTWARE, 4*, 870-877. doi:10.4304/jsw.7.4.870-877

[10] Yuan, C., He, Y., He, J., & Zhou, Z. (n.d). *A Verifiable Formal Specification for RBAC Model with Constraints of Separation of Duty*, 205-219. Retrieved from http://f3.tiera.ru/2/Cs_Computer%20science/CsLn_Lecture%20notes/I/Information%20Security%20and%20Cryptology,%202%20conf.,%20Inscrypt%202006(LNCS4318,%20Springer,%202006)(ISBN%203540496084)(313s).pdf#page=205

[11] Zhao, L. (2008). *A Role-based Access Control Security Model for Workflow Management System in an E-healthcare Enterprise* (Doctoral dissertation, UNIVERSITY COLLEGE)