Business Process Driven Framework for defining an Access Control Service based on Roles and Rules

Ramaswamy Chandramouli
Computer Security Division, ITL
NIST, Gaithersburg, MD 20899
(chandramouli@nist.gov)

Abstract

Defining an Access Control Service for an enterprise application requires the choice of an access control model and a process for formulation of access decision rules to be used by the access enforcement mechanism. In this paper, we describe a business process driven framework (called the BPD-ACS) for developing both the model and formulating the access decision rules. The model used is the Role Based Access Control (RBAC) model and the access decision rules are based on temporal business associations. The enterprise setting is a multi-facility hospital and the particular application for which the access control service was defined is the Hospital-based Laboratory Information System. (HLIS).The lesson learnt from this exercise is that a much more sophisticated rule processing capability is required for these types of applications than is currently available in both commercial and research-prototype authorization servers.

1. Introduction

Defining an Access Control Service for an enterprise application is rarely followed as a formal process. Very often the access control mechanism is chosen first (because of platform constraints) and the existing access control requirements for the application are often adjusted to fit into the limitations of the mechanism at hand. In situations where the organization's existing IT infrastructure does not pose the implementation constraints for access control, we can define the requirements for an Access Control Service through a relatively more formal process just like the determination of functional requirements for a a new software product.

In this paper, we propose a framework which we call the Business Process Driven Access Control Service (BPD-ACS) where each service component is defined based on a top-down analysis of the business processes that a given application is intended to support.

To define an access control service for any application requires an analysis of application-level operations, and the mapping of enterprise access control policies to those operations to derive a set of operational constraints governing those operations. Next a profile of the user base has to be determined and based upon the operational constraints, a modeling scheme for capturing user-operation interactions must be determined. These modeling schemes are what are commonly called the Access Control Models in the security literature.

The choice of an Access Control Model is dictated by factors like administrative convenience, policy support capabilities and in some instances the access control mechanisms available in the enterprise platforms. In our access control service framework, we have chosen the Role Based Access Control (RBAC) model [FCK95]. RBAC was chosen because of the administrative convenience it provides through the concept of roles which can be used for encapsulating all the application-level operations associated with a specific business process. Also, the support for RBAC is available on a wide variety of platforms like DBMSs and Operating Systems. Further, RBAC provides a taxonomy of models [SCFY96] of increasing complexity (and hence functionality) and any one of these models can be chosen depending upon the complexity of user-operation interactions.

However, many Access Control Models (including RBAC) have limitations with respect to fully representing all the facets of user-operation interactions. The two main facets are:

(a) A set of application-level operations (e.g., Creating a Purchase Request) a designated user is entitled to perform by virtue of his/her job function or organizational role. These are commonly called Privileges.

(b) Restrictions on exercising those privileges because of environmental variables like time of access or the application state. The information governing such restrictions are called contextual information and the restrictions themselves are called Access Decision Rules.

Most Access Control Models provide constructs for managing user-privilege association (by providing the ability to express privileges at different levels of granularity) but not necessarily in representing and processing Access Decision Rules. Hence there may be situations, in defining an overall Access Control Service, a separate service definition may be needed for representation and processing of Access Decision Rules.

Based on the above discussion, the salient features of the Business Process Driven Access Control Service (BPD-ACS) can now be outlined as follows:

(a) Definition of application-level operations based on Business Process Analysis
(b) Protection Requirements for those operations based on enterprise security policies
(c) Developing the RBAC model for the application
(d) Formulating, Representing and Processing Access Decision Rules.

The organization of this paper is as follows. In Section 2, we compare our work with related work in the area of RBAC modeling and Rule processing. In Section 3 we state the sequence of steps needed to define an access control service under the BPD-ACS framework. Sections 4 through 8 describe the operation of these steps to fully define an Access Control Service for a Hospital-based Laboratory Information System (HLIS). The concluding section (Section 9) presents other potential application domains where the framework could be applied and the scope for future work.

2. Comparison with Related Work

The main emphasis in the BPD-ACS framework is to chose an access control model that is relatively simple to administer, but at the same time address the business process requirements and policy restrictions through a flexible rule formulation and processing approach. Hence we have chosen the $RBAC_2$ model (roles with hierarchies) from the taxonomy presented in [SCFY96].

As regards rule processing capability, many ideas have been proposed to incorporate access decision rules within the framework of an RBAC model itself. Didriksen [D97] has used the concept of fragments to define predicates restricting the user access to a subset of rows and columns within relational database tables. These predicates (which can provide a more finer level of granularity than traditional database views) are then compiled into database triggers which will then enforce access restrictions at run-time. But the limitation of this approach is that it can only be used for enforcing access control rules for data stored in relational database tables and not for any general type of target objects. Giuri and Iglio [GI97] have proposed the concept of role templates which will encapsulate what they call parameterized privileges. These parameterized privileges are triples consisting of: (1) the access mode, (2) the target object and (3) a logical expression that will be evaluated for access on any element of the target object. But the biggest limitation in their approach is that the parameters in a role template are the same as those in each of the privileges they contain. This approach to role definition creates a tight coupling between the types of privileges a role can hold and may not be realistic in many application environments where each individual privilege in a role may have several different parameters.

A much more flexible approach for associating access decision rules with roles has been adopted in the HP Praesidium Authorization Server* [HPAS-WP] and in the Open Group's Research prototype Adage [AD-SO, BY] for distributed access control service (which is also based on the HP approach). In the HP model, rules are first defined independently of the roles (profiles) since they carry the same semantics as a privilege. The rule definitions contain boolean expressions comparing two sets of attributes – transaction attributes and privilege attributes. One or more of these rules are then referenced within a role (profile) definition and the authorized values for the privilege attributes are also specified there. Whenever a user performs a transaction using a set of values for transaction attributes which are used in a rule, the rule expression is instantiated (and hence evaluated) by obtaining the values for the privilege attributes specified in the user's role. The user is allowed to execute the transaction if the rule expression evaluates to true.

* Certain commercial products are mentioned in this paper. This does not imply recommendation or endorsement by the National Institute of Standards and Technology nor does it imply that the products mentioned are necessarily the best available for the purpose.

The Access Decision Rule processing approach outlined in this paper builds on the HP model and adds enhanced functionality. In the HP approach, the privilege attribute values are determined based on business rules which will change only when the rules are changed. On the other hand, in healthcare environments like hospitals where the association between entities specified in a transaction is very much short-lived (a patient in a ward and the attending physician for that ward), checking the validity of transaction attribute values (a Physician ID and Patient ID) cannot be performed using static privilege attribute values (Dr. John is authorized to order a test for Mr. David – the patient) specified in a role. In our approach the values needed for instantiating (evaluation) a rule are obtained dynamically at access decision time from a database which contains these types of temporal business associations.

3. Processing Steps in BPD-ACS Framework

The Business Process Driven Access Control Service framework consist of the following sequence of steps:

(STEP 1) For a given application system identify the business processes that it is intended to support. Also identify the information objects and methods to support the business processes. The output of this step will be the application-level operations.

(STEP 2) Determine access control requirements for each of the application-level operations. This step will be driven by the requirements in the enterprise access control policies. The output of this step will result in a set of application-level privileges and a set of constraints that govern the exercise of those privileges.

(STEP 3) Map the categories of users who are going to interact with the application and the corresponding methods in the application they need privileges based on their authorized business processes. This user-privilege association (based on the data obtained from STEP 2) will be modeled using the Role Based Access Control model.

(STEP 4) Again using the constraints (obtained from STEP 2), formulate a set of Access Decision Rules. Based on the data that is used in the rule’s predicates, define the supporting data for instantiating (evaluating) the access decision rules. This data will be housed in a database called the Temporal Business Association Database.

(STEP 5) Define the Access Enforcement Mechanism based on the access service components defined in STEP 3 and STEP 4.

A realization scenario for the above steps for a Hospital-based Laboratory Information System (HLIS) is described in the following sections.

4. Business Processes & Supporting Methods in HLIS (STEP 1)

The overall function of a Laboratory Information Systems (LIS) in healthcare settings is the storage and dissemination of information pertaining to various clinical tests performed on patients for the purpose of diagnosis and treatments. This application system may be deployed at various types of settings like a multi-facility hospital or in independent clinical service laboratories which provide clinical test services to various hospitals and physician groups. Though the major functionality may be common between LISs deployed at different settings, there may be differences in terms of interface and security requirements. For the purpose of guiding our discussion in this paper, we have chosen a Hospital-based Laboratory Information System (HLIS).

From an analysis of baseline functionality found in commercial offerings of HLIS, we found that it supports the following business processes.

(a) Lab Order Entry (Test Request)
(b) Lab Test Scheduling
(c) Capture and Recording of Test Results
(d) Quality Control checks on Test Results
(e) Generation of Summary Reports (if needed)
(f) Retrieve/Access Test Results

After the determination of the business processes to be supported by the application system, it is necessary to identify the broad information categories (or information domains) and the specific information objects within these information domains that are required to support a given business process. This requires an understanding of the overall functions involved in a business process. For example, the Lab Order Entry process involves a physician (or any authorized person) retrieving a patient basic demographic and insurance information and submitting a list of tests to be performed (from a set of valid tests that the lab is equipped to perform) on the patient. In terms of broad information domains, this process may require the following:

(a) Patient Information Domain
(b) Order Information Domain
(c) Procedure Codes Domain

An information domain consists of a broad class of related objects and every object in the domain may not be used in a particular business process. For example, the patient information domain may consist of objects that stand for patient insurance information, patient demographic information, patient encounter information, patient treatment regimens, patient allergies etc. The information objects from the patient information domain that may be needed for submitting a lab test request may just consist of those dealing with insurance and demographic information and the patient’s current location. By using a similar analysis, information objects from other domains may be determined and the overall set of information objects needed for supporting the Lab Order Entry business process is derived as follows:

(a) Patient Demographic Information Object
(b) Patient Location Information Object
(c) Patient Insurance Information Object
(d) Order Header Information Object
(e) Order Work-List Object
(f) Lab Test Codes Object

Any business process analysis requires a determination of both data and functional requirements. We have now completed the the data requirement component for the Lab Order Entry business process. The functional requirements for this process are defined in terms of the following abstract method definitions. These form the HLIS application-level operations for the Lab Order Entry business process.

(M1) Get_Demo_Info(PatientId, AccessorId)
(M2) Get_Location_Info(PatientId,AccessorId)
(M3) Get_Insurance_Info(PatientId, AccessorId)
(M4) Set_Test_Request (PatientId,PhysicianId,AccessorId)
(M5) Set_Work_List(OrderId,AccessorId)
(M6) Get_Lab_Codes (AccessorId)

A brief explanation regarding the intended functionality of the above defined methods is as follows:

(M1)Get_Demographic_info ( ) - gets demographic information regarding a patient like the Patient Name, Age/Sex/Race/DOB etc. Physically this information may be residing on an another system and the right to invoke this function is determined based parameter values like PatientId and AccessorId. This right is determined based on the fact whether a "need-to-know" situation exists for the AccessorId to retrieve the demographic information of the patient represented by PatientId.

(M2) Get_Location_Info() – gets information about the location in the hospital where the patient is undergoing treatement. This may consists of Type_of_Ward, Ward Number, Room Number, etc.

(M3) Get_Insurance_Info() - retrieves information like the Insurance Company Name & Address, the Policy/Group No, the name of the Primary Insured, etc. for the patient for whom the tests are requested.

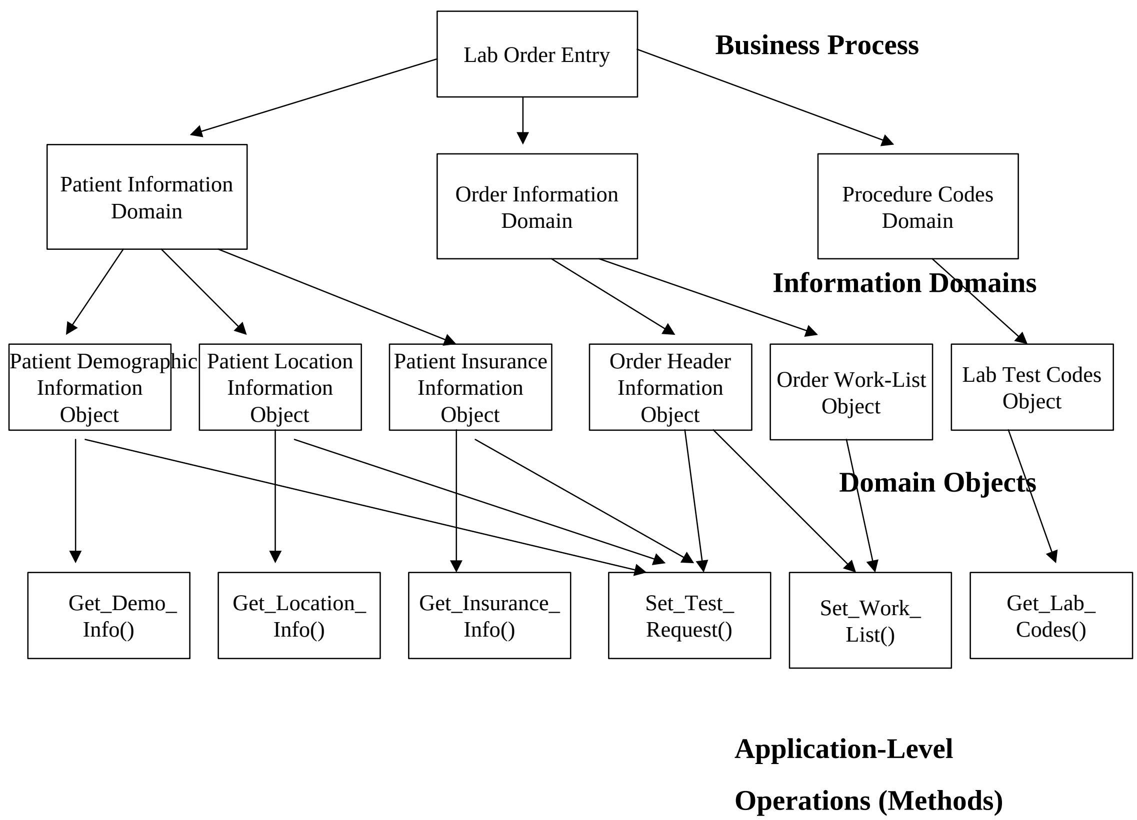
(M4) Set_Test_Request() – Provides all the necessary information about the patient (like the basic demographic information, current location and insurance information) and the overall type of test requested, test request date/time, test priority, etc.

(M5) Set_Work_List() - may set information regarding the line items in a test request representing each type of test. The right to order a particular kind of tests depends upon the physician making the requests. For example certain tests could only be ordered by a Specialist rather than by a General Physician.

(M6)Get_Lab_Codes() - displays the list of tests that the lab is equipped to perform and the standard codes associated with each of them.

A schematic diagram depicting the derivation of application-level operations for a given business process is shown in Fig 4.1.

Fig 4.1 Deriving Application-Level Operations from a Business Process



5. Mapping Enterprise Security Policies to HLIS Operations (STEP 2)

The next step after determination of application-level operations is to look at the provisions in the enterprise security policy (specifically enterprise access control policy) that will have an impact on those operations. In general, an enterprise access control policy may be a synthesis of the following information categories:

(a) Enterprise best practices (which has evolved over a period of time)
(b) Threat model driven requirements
(c) Government Regulations (Federal, State, etc.)

Because of the diverse sources based upon which security policies are composed, these may sometimes be stated at different levels of granularity. For example, the requirements in Government regulations (like HIPAA) may state access control requirements in terms of the information domains affected by application-level operations. Internally developed operational manuals may state access control restrictions in terms of the healthcare worker categories within the hospital. Hence, mapping enterprise access control policy requirements to application-level operations may sometimes turn out to be a non-trivial task.

Let us now examine our HLIS application operation M4 [(Set_Test_Request ()]. As could be seen from Fig 4.1 this operation uses information from Patient Demographic Information Object, Patient Location Object (which contains information about the ward in which the patient is staying, admittance date, etc.) Patient Insurance Information Object and Order Header Information object. The combination of the information contained in these objects may come under the category of *Health Information* as per the definition in the *Security and Electronic Signature Standards; Proposed Rule* issued by Health and Human Service (HHS) [SESS98]. As per the proposed regulation, the "health information means any information …… that relates to past, present, or future physical or mental health or condition of an individual…". With respect to our method M4, the patient location information and the kind of tests ordered can reveal the condition of the patient and hence this is a restricted class of information that can only be created by authorized individuals. Based upon this federal regulation, the hospital access control policy can stipulate that creation of laboratory test request information can only be created by the patient's attending physician or a Registered Nurse authorized by the attending physician. Hence the HLIS access control requirement for Set_Test_Request() can be stated as follows:

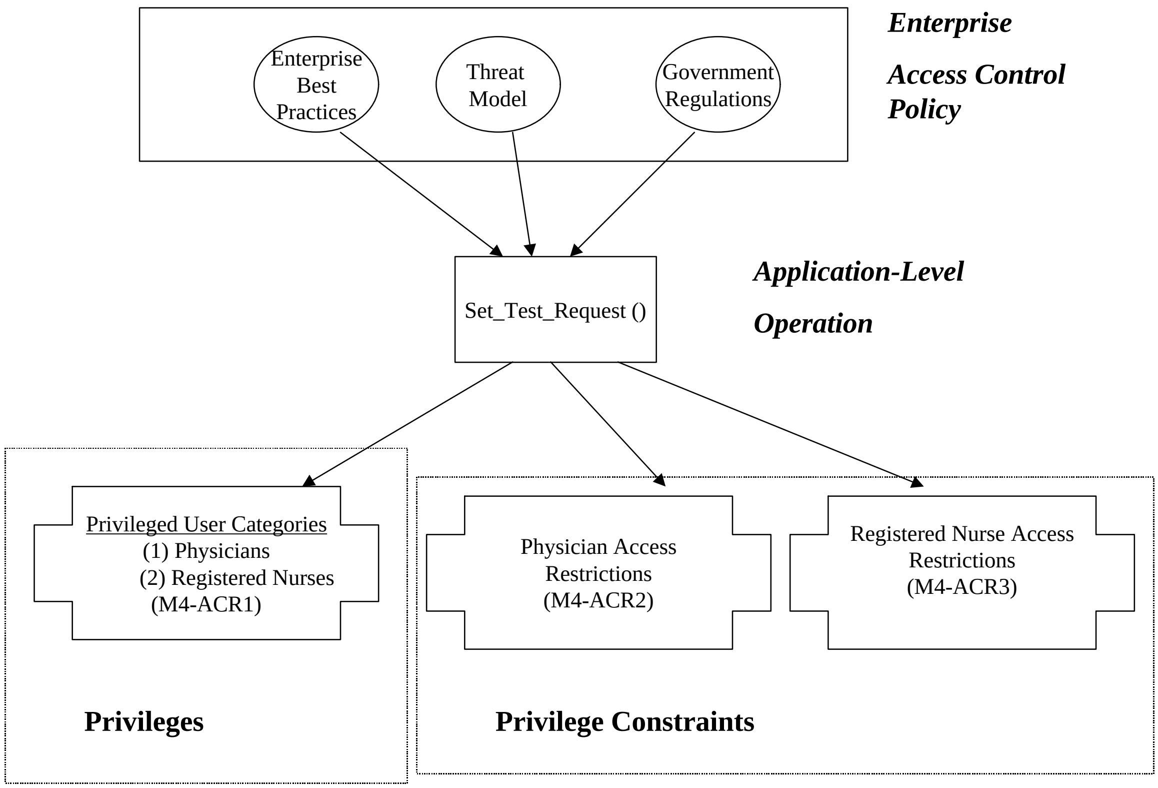
(M4-ACR1) Only the following categories of healthcare workers are authorized to order laboratory tests – Physicians and Authorized Registered Nurses

(M4- ACR2) The Physician ordering the laboratory test must be the current attending physician of the patient.

(M4 – ACR3) The Registered Nurse ordering the laboratory test must be one of those authorized for ordering tests either by the attending physician or by the hospital administration based on a trust policy.

A schematic diagram for mapping organizational enterprise access control policy requirements to access control requirements for HLIS application-level operation Set_Test_Request (PatientId, PhysicianId, AccessorId) is depicted in Fig 5.1.

Fig 5.1 Privileged Users and Privilege Restrictions for a HLIS application-level operation Set_Test_Request ()



6. Defining the Access Control Model for the Application (STEP 3)

Since we had chosen RBAC as the modeling foundation for our application, we have to determine the members of the three broad entities that RBAC model uses. These entities are Users, Roles and Privileges.

Let us now focus our attention on the Privileges first. Recall that in the last section, our analysis of a business process (Lab Order Entry) supported by the application resulted in a set of methods (M1 through M6). If we had made an architectural decision that all the user interactions with information objects used in an application are going to be through a set of methods, and not through any other lower level access on the object contents, then the methods themselves can form the correct level of granularity for defining the privileges.

The next design aspect for building the RBAC model for the HLIS application is the definition of roles. In RBAC literature, roles are generally defined to group together all the privileges associated with a job function or a business process. Since we have viewed our HLIS as one supporting a fixed set of business processes, and a meaningful interaction of any user with the application can only take place if the user can perform at least one business process, it makes sense to group all the privileges together based on a business process. Based on this logic, we can group together the methods M1 through M6 that were found required for supporting the Lab Order Entry business process and assign to a role which we shall call the "Test_Requester" role.

The last task remaining is to identify the members of the User entity. An hospital environment just like any other enterprise consists of groups of people performing the same business tasks and being subjected to a common policy governing access privileges from a user to a process. Some of these groups are Physicians, Nurses, Lab Technicians, Pharmacy staff, Hospital Administrators, etc. The common business processes that a particular group (say Physicians) perform are (a) examining patients (b) prescribing medications or treatment programs and (c) ordering clinical tests and analyzing the results. Various names like Group, Team, and Trusted Access Domain are given in the access control literature for the collection of users. In our BPD-ACS framework we will use the term Trusted Access Domain (TAD). Thus we will have Physician TAD, Nurse TAD, Administrator TAD, etc.

Since the semantics behind the grouping of users is the similarity of business processes they perform and since the roles have been defined as a grouping construct for all the privileges associated with a business process, it becomes natural to associate roles with the TADs. The mapping of these Trusted Access Domains in the Hospital environment to the HLIS application roles are given in Table 5.1 below. A portion of the schematic diagram of the RBAC model for HLIS application is shown in Fig 5.2.

Table 5.1 - Map of Trusted Access Domain to HLIS Application Roles

Hospital Trusted Access Domains	***HLIS Application Roles***
General Physician	Test_Requester, Report_Viewer
Speciality Physician	Test_Requester, Report_Viewer
Lab Supervisor	Test_Scheduler,Results_QC
Lab Technician	Test_Results_Generator
Registered Nurse	Test_Requester, Report_Viewer

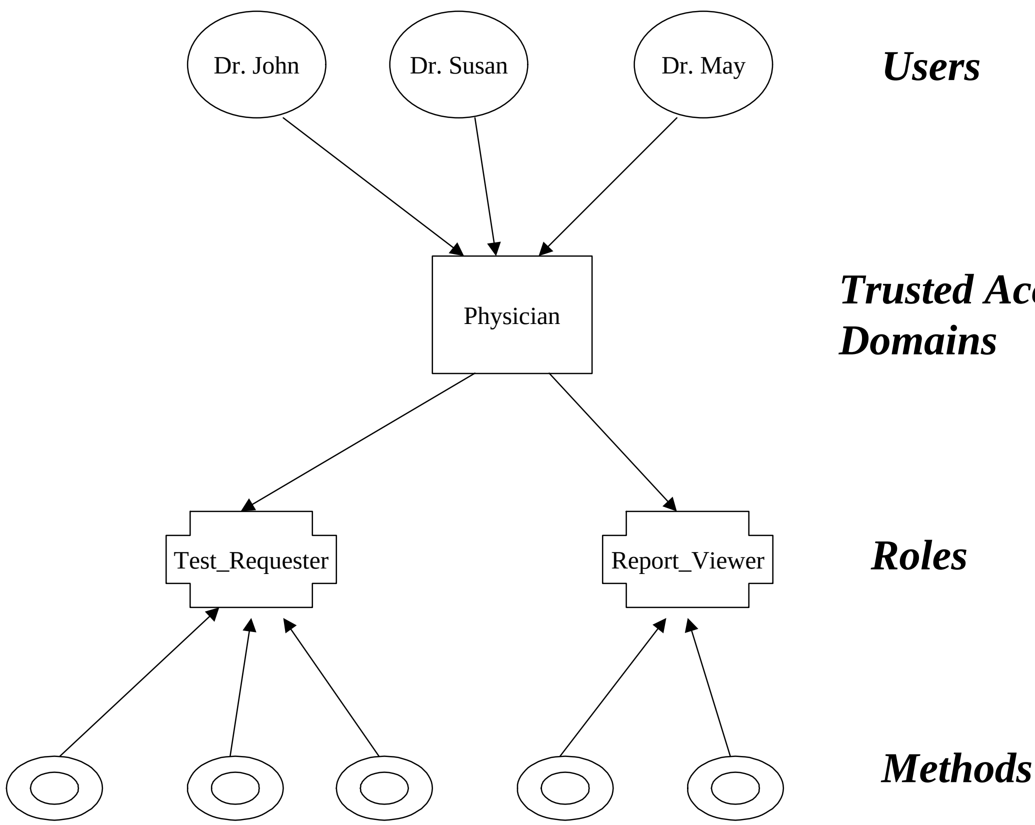
7. Definition of Access Decision Rules (STEP 4)

Access Decision Rules constrain the exercise of privileges. They are meant to bring in contextual information to decide whether a given privilege can be exercised in a particular access request. Depending upon the type of contextual information they use, they can made up of different types of constraints:

(a) Time/Day of Access Request (Time Constraints)
(b) History of Previous accesses (Conflict of Interest Constraints)

(c) History of Previous accesses together current state of the target object (Separation of Duty Constraints)
(d) Trust Level of the User (Trust Constraints)
(e) Parameter values used in the access request (Temporal Business Association Constraints)

Fig 5.2 – A partial schema of the RBAC Model for HLIS Application



Each of these types of access decision rules are meant to implement different kinds of policies. In Healthcare enterprises like Hospitals, the predominant factors constraining the exercise of privileges are the associations or relationships between the providers of service (like physicians, nurses, etc.) and the recipients of service (in-patients, out-patients, etc.). Because these associations are relatively short-lived, the most significant constraints are the Temporal Business Association Constraints. Hence access decision rules involving only this type of constraints are discussed in this paper though our BPD-ACS framework itself is meant to support any type of rule.

Access Decision Rules involving temporal business association constraints compare the parameter values in a given access request (based on an assigned privilege) to the data values that will validate the current business association. These validating data values are present in what we call as the Temporal Business Association Database. Hence the rule predicate expressions consists of relations which link the access request parameter names to column names found in the Temporal Business Association database through appropriate relational operators like ==, <=, >=, etc. Hence the truth values of these predicate expressions can only be obtained by instantiating these access decision rules by retrieving values from the Temporal Business Association database.

The definition of an Access Control Rule based on the syntax used in [ASG99] that will constrain the access to method Set_Test_Request (PatientId, PhysicianId, AccessorId) by a Physician or Authorized Nurse for the purpose of creating a Laboratory Test Request is shown below:

Listing 7.1 - Definition of the Allow_Set_Test_Request rule for constraining the Privilege Set_Test_Request (PatientId, PhysicianId, AccessorId)

```
Rule Name
        Allow_Set_Test_Request

Access Request Attributes
        PatientId:  string
     PhysicianId:  string
     AccessorId:  string

Environmental Attributes
  Accessor_Domain: string

Temporal Business Association Database Attributes
            Table_Name:  ATTENDING_CLINICIAN
            Field_Names:
                  Patient_Identifier: string;
             Physician_Identifier: string;
          Auth_Nurse_Identifier:  string;

Rule Predicate
                  PatientId  = = :Patient_Identifier &
    (( Accessor_Domain  =  "Physician" & PhysicianId  = =  :Physician_Identifier) |
     (Accessor_Domain  =  "Nurse" & AccessorId  = =  :Auth_Nurse_Identifier ))
```

(NOTE: the : in front of the field names (e.g., :Patient_Identifier denotes that it represents the actual value from the database and not the name of the column or attribute from the ATTENDING_CLINICIAN table).

After defining the rule Allow_Set_Test_Request, it must be associated with the corresponding privilege (or method) Set_Test_Request() wherever the invocation of this method is to be restricted. This method may be directly assigned to a user as a privilege or indirectly to a role. In every one of these assignments, the association of this rule to the method must be represented. But in situations where some user or role is to be granted unrestricted privilege for this method, this rule association need not be shown. The association of this rule Allow_Set_Test_Request to the method Set_Test_Request() when it is assigned as one of the privileges in the role Test_Requester is shown below (in bold letters) in Listing 7.2:

Listing 7.2 – Role Definition showing the methods and associated rules

```
Role Name = "Test_Requester"
Role Memberships = <none>  /* Here memberships means other roles –
                              not users */
Privileges:

Privilege Name = Get_Demo_Info(PatientId,AccessorId)
Privilege Rules:
    Rule Name: Allow_Get_Demo_info

Privilege Name = Get_Location_Info(PatientId,AccessorId)
Privilege Rules:
    Rule Name: Allow_Get_Location_info

Privilege Name = Get_Insurance_Info(PatientId,AccessorId)
Privilege Rules:
    Rule Name: Allow_Get_Demo_info

Privilege Name = Set_Test_Request (PatientId,PhysicianId,AccessorId)
Privilege Rules:
    Rule Name: Allow_Set_Test_Request

Privilege Name = Set_Work_List (OrderId,AccessorId)
Privilege Rules:
    Rule Name: Allow_Set_Work_List

Privilege Name = Get_Lab_Codes (AccessorId)
Privilege Rules:
    Rule Name: Allow_Get_Lab_Codes
```

From the above definition of the role Test_Requester, we see that all the methods that we had earlier mentioned as comprising this role are assigned as privileges with associated rules for restricting the exercise of each of these privileges. The above syntax will also permit definition of multiple rules for the same method as well.

Having shown the association of an access decision rule to a method (or privilege), we will now illustrate with an example the conditions under which a given access request will be granted. Supposing that the user request to the method Set_Test_Requester() consists of one of the following attribute values.

Set_Test_Request("P102068","MD23456","MD23456") or
Set_Test_Request(("P102068","MD23456","RN8967")

(The first request example may be from the physician in which case the Accessor_Domain == "Physician" and the second request may be from a Registered Nurse in which case the Accessor_Domain = "Nurse").

Then the rule expression in Listing 7.1 will only evaluate to true if the following entry (Listing 7.3) is found in the ATTENDING_CLINICIAN table in the Temporal Business Association Database.

Listing 7.3 – A Table in the Temporal Business Association Database

Table Name: ATTENDING_CLINICIAN

Patient_Identifier	Physician_Identifier	Auth_Nurse_Identifier
P102068	MD23456	RN8967

8. Defining the Access Enforcement Mechanism (Step 5)

The process of defining the access enforcement mechanism consists of providing the logical sequence of steps involved in arriving at an access decision for a given access request. We will illustrate this process for the case where a user called Dr. John makes a request for ordering laboratory tests for a patient by name David through the HLIS application [(Invoke the method Set_Test_Request() with actual parameters (DavidId, JohnId, JohnId)].The Access Request Processing (ARP) steps based on the access service components defined in BPD-ACS follows:

(ARP-1) After the user (Dr. John) is authenticated, his trusted access domain was determined to be the General Physician. This information is obtained by consulting the database Trusted Access Domain (TAD) – DB.

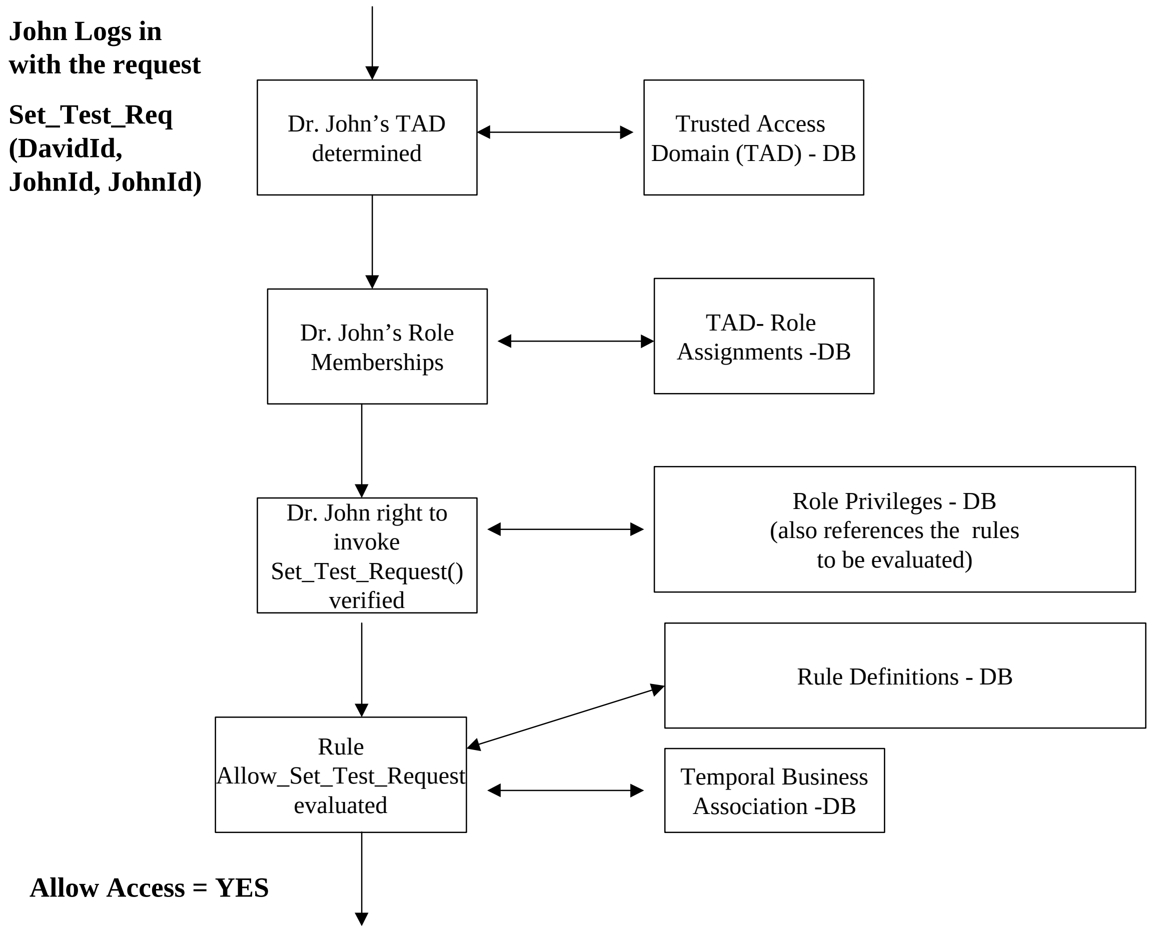
(ARP-2) The HLIS application roles that are assigned to this domain are the Test_Requester and Report_Viewer (Table 5.1). These role assignments are obtained by consulting the TAD-Role Assignments DB.

(ARP-3) The privilege domain for Dr. John consists of the combination of all the methods in these two roles . Since Set_Test_Request() is one of these methods the system continues to process Dr. John's access request (this information is obtained from the Role Privileges DB). The system also obtains the rule (Allow_Set_Test_Request) to which needs to be evaluated to determine whether John can submit a lab test request for David.

(ARP-4) The rule Allow_Set_Test_Request is evaluated by obtaining the rule definition from Rule Definition DB and the validating values from Temporal Business Association DB. If John is the attending physician for David then John's access request Set_Test_Request(DavidId, JohnId, JohnId) will be allowed.

The schematic diagram of the above described sequence of steps is shown in Fig 8.1.

Fig 8.1 – Access Decision Logic in the BPD-ACS framework



9. Conclusions

The Formulation of Access Decision Rules based on temporal business association constraints, which was illustrated in the context of an healthcare enterprise application, could also be potentially be used in other application domains. One promising area is in extranet applications where the business relationships between enterprises are of a short duration. These types of rules could also be used in web-based bidding and auction applications where the rights of the interacting parties are determined based on occurrence of certain events and current state of relationships among the interacting parties.

Further, the underlying access control model itself may contain various constraints that need to be procedurally supported. For example, the RBAC model used in our framework could contain constraints associated with user-role assignments, user-role activations and privilege-role assignments. The BPD-ACS framework could be enhanced to include these constraints through various Model Based Rules in addition to Access Decision Rules.

Thus we see that the BPD-ACS framework could be scaled up to include more access control service components depending upon the application domain needs. But the underlying theme is that each of these service component definitions should be based on a sound analysis of the business processes that the application is intended to support.

10. References

[AD_SO] "Adage System Overview", http://www/.memesoft.com/ adage/docs/SystemSpec.ps

[ASG99] HP Authorization Server Guide, October 99, Hewlett-Packard Company, Palo Alto, CA.

[BY] W.R. Bevier and W.D.Young. "A Constraint Language for Adage" http://www.memesoft.com/adage/al.ps

[D97] Tor Didriksen "Role Based Database Access Control – A Practical Approach", Proceedings of the 2nd ACM workshop on Role Based Access Control, November 1997, p 143-151

[GI97] Luigi Giuri and Petro Iglio "Role Templates for Content-Based Access Control", Proceedings of the 2nd ACM workshop on Role Based Access Control, November 1997, p 153-159

[FCK95] D.Ferraiolo, J.Cugini, and D.R.Kuhn. "Role Based Access Control (RBAC): Features and Motivations" Proc. 1995 Computer Security Applications Conference, December 1995, p241-248.

[HPAS_WP] HP Praesidium / Authorization Server White Paper. http://www/hp.com/security/products/authorization_server/papers/whitepaper/.

[SCFY96] R.S. Sandhu, E.J.Coyne, H.L.Feinstein and C.E.Youman. "Role Based Access Control Models" IEEE Computer, vol 29, Num 2, February 1996, p38-47.

[SESS98] Security and Electronic Signature Standards; Proposed Rule. Federal Register, Vol 63, No. 155, August 12, 1998.