POLITECNICO DI MILANO
MANAGEMENT ENGINEERING DEPARTMENT

POLO TERRITORIALE DI COMO
MASTER OF SCIENCE IN MANAGEMENT ENGINEERING

FROM RBAC TO NEW ACCESS CONTROL MODELS; THE CASE OF ABAC AND UCON

Advisor: Mariagrazia Fugini

Master's Degree Graduation Project

Jaime Alberto Aluja León – 797613

Academic Year 2013/2014

Acknowledgements

For their patience and understanding I would like to thank to
My mom, my dad and my brother,
Prof. Mariagrazia Fugini
Ms. Viviana Pulido Franco
And to all the friends I have made along the way.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLE

ABSTRACT

With an increased need for more thorough Access Control Methods for Information Systems (IS), some alternatives to the already existing methods have been designed to better handle the distribution of the IS, the integrations with customers and suppliers, giving a finer-grained approach to the security requirements of firms while maintaining the information inside the system secure. This work aims to compare the administrative features of the standard Role-Based Access Control (RBAC) model, against the Attribute-Based Access Control (ABAC) and Usage Control (UCON) models. The models are developed to explore how would an RBAC implementation on a firm and its administrative model would change if the IS used ABAC or UCON as its Access Control Methods. Results show the impact that the change might have in a firm.

Keywords: Access Control, RBAC, ABAC, UCON, Information Systems, Information Systems Administration.

SOMMARIO

Con una maggiore necessità di metodi più accurati nel controllo di accesso per i Sistemi Informativi (SI), alcune alternative ai metodi già esistenti sono stati progettati per gestire meglio la distribuzione dei SI, le integrazioni con clienti e fornitori, dando un approccio più dettagliato per i requisiti di sicurezza delle imprese, mantenendo le informazioni all'interno del sistema sicure. Questo lavoro si propone di confrontare le caratteristiche amministrative del modello standard di controllo di accesso basato sui ruoli (Role-Based Access Control - RBAC), contro i modelli di controllo di accesso basato su gli attributi (Attribute-Based Access Control - ABAC) e il controllo d'uso (Usage Control - UCON). I modelli si sviluppano per esplorare come cambierebbe un'implementazione RBAC di un'azienda se vengono utilizzati ABAC o UCON come metodi di controllo di accesso. I risultati mostrano I possibili impatti che la modifica potrebbe avere in un'azienda.

Parole chiave: Access Control, RBAC, ABAC, UCON, Sistemi Informativi, Gestione di Sistemi Informativi.

INTRODUCTION

As access to information becomes more pervasive, the task of keeping the information secure becomes more difficult. Providing adequate levels of security of information in the different types of Information Systems used by firms, and people is one of the most challenging tasks that management can encounter.

This difficulty comes as the users of Information Systems are diversifying and so are their Security needs; this is one of the reasons why, in lieu of a universal Access Control model, organizations and firms in private and public sectors are building ad-hoc solutions that best fit the security needs of their clients (or themselves) (Park & Sandhu, 2004).

An Access Control Method is then, a tool that helps to limit and restrict the activities of users within an Information System (IS). But, as it is the case in almost every firm, not all the users in the IS are meant to have the same access level to the available information. Moreover, this can create a breach in the Information Security System, as the intern may leave the firm with critical information.

In other words, Access Control serves as a checkpoint to ensure that only the right people have access to critical information inside a firm, and to ensure that they only have the right to Create, Read, Modify or Delete information in the IS if they have been qualified to do so.

Some of the already available solutions have gained such a wide acceptance in the Information Systems Security world that have become standards (Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001); as they grant proper, sufficient and timely access to the objects (or resources) in their domain.

Although highly effective, there has been an issue with AC models up until now, as the information changes from one owner to another (as is the case with music purchased online) the issuing firm has no control over who can access the information given away. Methods such as Digital Rights Management (DRM) have appeared as the answer to provide the information with a reference monitor of activity from the Client's side (Park & Sandhu, 2004). The need to integrate Access-Control models in the firm's Information Systems, with the information delivered to a client outside the firm is now one of the biggest issues regarding access to sensible information.

But, although there are more methods to secure information than ever, issues regarding feasibility, scalability, performance, and costs need to be taken into account.

PROBLEM DESCRIPTION

Guaranteeing trusted access to information, and guaranteeing the integrity of the information inside a secured information system has been a central paradigm in the history if Security of Information Systems; and given the rising amount of information shared between users (and their devices) and their surroundings, an opportunity exists to use devices as a method to retrieve information regarding the status of the surroundings among us in any type of event or situation. If used properly, information exchanges between our environment and us can be beneficial for users, urban developers, building administrators, security forces, etc.

However, any benefit gained from the exchange of information can be hampered if the information sent back and forth between the users and the environment is incomplete, misleading, or fake.

As the role of Information Systems has changed, so have the methods to control the access to information. Models such as Mandatory Access Control (MAC) and Discretionary Access Control (DAC) were the norm, mainly due to the fact that MAC was used by the Department of Defense (DoD) (Ferraiolo & Kunh, 1992), but as applications spread to usage away from the intentions of the DoD, it seemed that the previously mentioned Access-Control methods were not well suited.

Attribute Based Access Control (ABAC) is another example of how the changes in the Information Systems have driven Access Control methods. Just as RBAC, and MAC and DAC before it, ABAC is an Access Control method that simplifies some instances of RBAC, but creates more complex interactions in other (Coyne & Weil, 2013).

As the access to digital resources is becoming more and more ubiquitous (via Smartphones, PDA's, eBook readers, Tablets, and internet-integrated home

appliances, among others), new challenges to ensure a trusted and controlled access to these resources have risen (Park & Sandhu, 2004).

PROBLEM JUSTIFICATION

As stated before, this predicament has caused a lot of solutions (including ad-hoc solutions) to be created to address the aforementioned problem, as the early types of access control do not seem to suit well a dynamic environment (Ferraiolo & Kunh, 1992)such as that found in smart environments.

OBJECTIVES

GENERAL OBJECTIVE

Compare the Attribute-Based Access Control (ABAC), and Usage Control with their counterparts by evaluating their efficacy as possible Access-Control implementations in more complex, loosely coupled and more dynamic environments.

SPECIFIC OBJECTIVES

- Define the parameters by which the Access-Control models will be evaluated and compared.
- Evaluate the context in which the models were designed and their potential of expansion to other realms.

BIBLIOGRAPHIC RESEARCH

Before pointing out the existing differences between different Access Control methods, it is relevant to understand what the principles of Access Control are. As stated above, the function of Access Control Methods is to limit the activities that an identified user is allowed to perform within an IS, Access Control must mediate with every attempt of a user to exert an action in the objects found in the IS.

FUNDAMENTALS OF ACCESS CONTROL

ACCESS CONTROL MECHANISMS

An **Access Control Mechanism (ACM)** is nothing other than the logical components, deployed by IS Security Designers, Architects, and Administrators, that are used to protect the objects found in the IS. The protection is set by mediating request, and enforcing the access decisions from the subjects that want to perform an action on the objects inside the IS (Hu, Ferraiolo, Kuhn, Schnitzer, Miller, & Scarfone, 2014).

ACCESS CONTROL PRIMITIVES

Each example of an Access Control system has a large and different variety of security, and administrative features, these features may include proprietary functions, attributes and methods to configure a class of AC Policies. (Hu & Scarfone, 2012). As stated by (Hu & Scarfone, 2012) "instead of being individually managed, permissions of practical AC mechanisms are organized in terms of and derived from a set of policy specific user attributes, providing a strategy for organizing, managing, and reviewing permission data, and controlling the access requests of subjects".

These policies are a representation of rules and relationships, which allow to determine if a requested access should be allowed (Hu, Ferraiolo, Kuhn,

Schnitzer, Miller, & Scarfone, 2014), given the capabilities of a subject on the objects.

Access Control primitives are defined by Hu & Scarfone (2012) as:

> The common operands for AC attributes and functions of AC mechanisms. Abstractly, AC mechanisms apply a set of rules to system states for the purpose of allowing or denying a specified combination of primitives. The rule set is composed according to the AC policy, such that the final process of any AC is the decision making for an access request that matches a set of AC primitives.

AC primitives include Subject, Object, actions, capabilities and privileges, these primitives are established through System, or Administrative assignments, and are defined as follows.

A **subject**, in broad and simple terms is an active entity, it can be a person, an organization, or a Non-Person Entity (NPE) (Hu, et al., 2014), a subject is the entity that causes information to flow among objects, or changes the system state (Hu & Scarfone, 2012). An NPE is an application or server that accesses information on behalf of a person, an organization, or maybe even both.

An **object** (also referred to as a Resource) is an entity that needs protection from unauthorized usage or modification (Hu, Ferraiolo, Kuhn, Schnitzer, Miller, & Scarfone, 2014).It is a passive entity containing and receiving information. Accessing a resource may imply accessing the information it contains (Hu & Scarfone, 2012).

Actions are the active processes, which a subject is in capability to invoke (whether he "knows" these capabilities or not). Actions allow a subject to

modify the state of the objects, and refer to a specific operation applied to the objects (Hu & Scarfone, 2012).

Explained by Hu, et al. (2012), the **Capabilities** (whose existence depends on whether or not the AC system uses them) specify the actions of the subject. Usually arranged as a list, these correspond to a row in the Access Control Matrix (A concept that will be described further on). Technically speaking, the entries in the list (the capabilities), are pairs arranged as follows: <*set of actions, object*>. Depending on the AC system, users can give their capabilities to other users.

Security Services

As of this point, it should be clear that in order to exist, AC methods cannot exist on their own, not only their purpose must be aligned with the mission of a firm (Hu & Scarfone, 2012), but they form part of a bigger framework needed to achieve their function. So far we have established at least two other entities for the method to even exist, these are the subject and the object, but there are many more.

Ideally, other services that come together with AC methods are the Authentication Service, the Reference Monitor, the Authorization Database, the Administrator of the Database, and an Auditing Service (See Figure 1) (Sandhu & Samarati, 1994).

Explained by Sandhu, et al. (1994), the user first encounters the Authentication Service. The **Authentication Service** is responsible for establishing the identity of the user (usually in the form of the username and the password). Once the identity of the user has been established, that information is sent to the **Reference Monitor**, which checks the existence of the User, and the Rights it has within the IS against an **Authorization Database**. As the identity of the User is confirmed, the AC method decides which Policy applies to it, and enforces it upon the User in order to let it have

access to the object. Finally, **Auditing Services** serve to monitor and keep record of relevant activities in the IS.

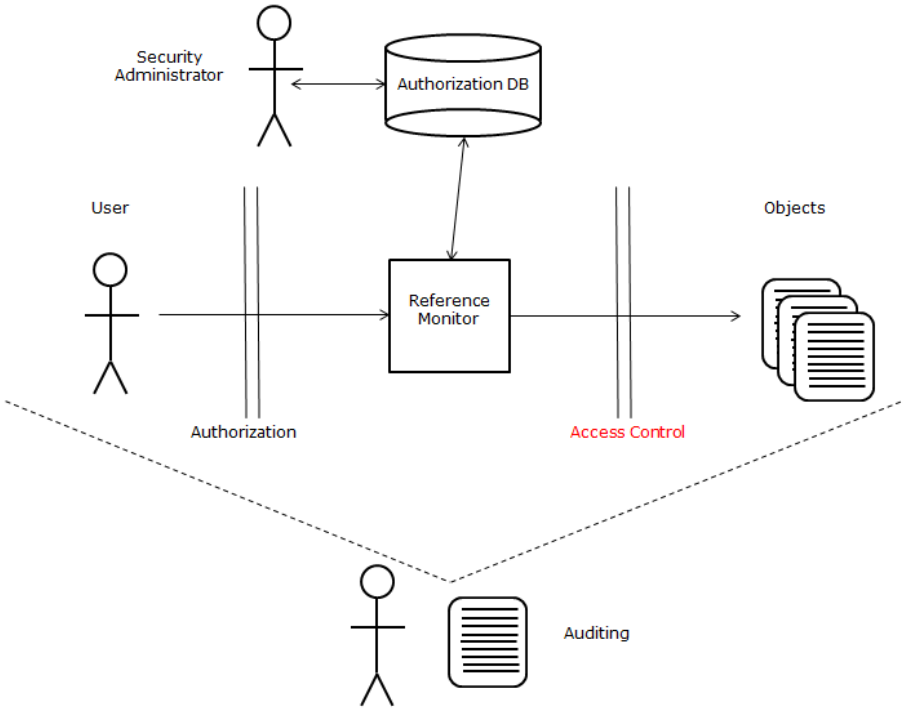

Figure 1. Security Services that accompany the AC method

Access-Control methods have been around for over three decades, and as times have changed, so have the rules determining access-control and digital rights to electronic resources.

Request-Response Paradigm

Before exploring the evolution, and the different types of AC methods, it is important to consider that most of them rely on the Request-Response Paradigm. In this paradigm, a Policy Enforcement Point (PEP) intercepts all subject requests to protected objects, and enforces policy decisions in response to them (Hu, Ferraiolo, Kuhn, Schnitzer, Miller, & Scarfone, 2014), the PEP serves as the intermediary between the Requests made by any Subject, and the Object (Coyne & Weil, 2013).

The PEP cannot grant or deny the Access to the protected resources on its own, the Policy Decision Point (PDP), whose function is to compute access decisions by evaluating the applicable policies for each access request, takes the AC decisions. One of the main functions of the PDP is to mediate and resolve conflicts between overlapping policies (Hu, Ferraiolo, Kuhn, Schnitzer, Miller, & Scarfone, 2014).

()serves as the retrieval source of attributes, or the data required for policy evaluation to provide the information needed by the PDP to make the decisions" (Hu, Ferraiolo, Kuhn, Schnitzer, Miller, & Scarfone, 2014).

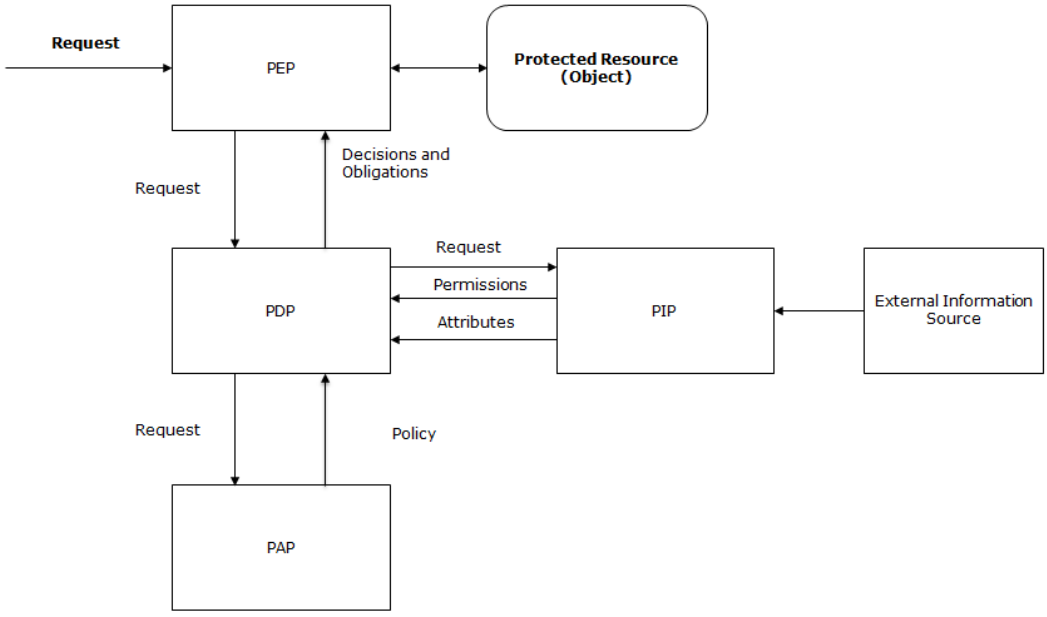

Figure 2. Graphical Representation of the Request-Response Paradigm

Access Control Matrix

The Access Control Matrix is a conceptual model, that specifies the actions that each subject can exert on each object on the IS (Sandhu & Samarati, 1994). Sandhu & Samarati also specify that each subject has a row in the

matrix, and each object has its corresponding column, the reference monitor is then responsible for mediating the interactions inside the IS.

Access Control List

It can be obvious at this point that keeping an Access Control Matrix in an expanding IS can be troublesome, as the number of subjects and objects increases. A simplified version of the Access Control Matrix is the Access Control List (ACL); Sandhu & Samarati explained that for each Object in the IS, there is an associated ACL, in it is possible to review the permissions that each Subject has on the Object. (Hu, Ferraiolo, Kuhn, Schnitzer, Miller, & Scarfone, 2014).

The privileges that any Subject has on the Object (read, write, delete, etc.), can be managed by the owner of the Object, and for each subject to be added to the ACL, the owner of the Object must first evaluate the identity of the subject, the Object, and the Attributes against the policies pertinent to the object, then the owner can decide whether or not add the subject to the ACL (Hu, Ferraiolo, Kuhn, Schnitzer, Miller, & Scarfone, 2014). Hu, et al. (2014) also state that the decision is static, and in order for an owner to reevaluate the existence of a subject in the ACL, a notification is required.

Hu, et al. (2014), also warn "Failure to remove or revoke access over time leads to users accumulating privileges".

Mandatory Access Control

Yuan, et al. (2005), define Mandatory Access Control (MAC), is a method by which access to objects is restricted based upon fixed security attributes (also known as labels) assigned to both Users (subjects), and objects. These attributes are mandatory, which means that they are enforced by the System's architecture and cannot be modified directly by users or any Non-

Person-Entity (NPE). As will be the case most of the time, only a system administrator is allowed to change the labels.

The need for MAC arises as a security policy in a firm might state that the owner of the object to be protected by the Access Control method is not entitled to grant access to the object by himself (or herself), such policy was the reason why the Department of Defense (DoD) implemented this type of Access Control. As a common rule, the labels on an object are called a security classification, whilst those same labels when applied to users are called security clearances (Yuan, et al., 2005).

Take as an example a document, which contains information labeled as classified. Someone with a security clearance 'X' might read it, as 'X' allows the subject to read Unclassified and Classified files might only read such a document; but, if the same user wants to access a document with a security classification of Secret or Top Secret, he (or she) cannot read the information contained in the file (Fugini, 2012). This is what is known as a "lattice".

Discretionary Access Control

Alongside MAC, Discretionary Access Control (DAC) serves as a method to restrict access to objects based in the identity of the user, the groups, and processes to which they belong, as well as his (or her) need to know about the information contained within the object. The access modes such as read, write, or execute are granted to a user if the user has privilege to use a specific access mode on an object (Park & Sandhu, 2004). The controls are discretionary in the sense that a subject with certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.

Unlike MAC, in most DAC literature, users and subjects are used interchangeably without clear distinctions in their definitions. The access

modes such as read, write, or execute are granted to a user if the user has privilege to use a specific access mode on an object. Either access control list (ACL) or capability list can be used for authorization rules (Park & Sandhu, 2002).

Note that MAC and DAC are not mutually exclusive. In fact, many systems use MAC and DAC in conjunction. In this case, mandatory controls are necessary but not sufficient conditions for the indicated access. They can be checked first, followed by a check of the discretionary controls. Alternatively, mandatory controls can kick in after the discretionary checks are satisfied.

There are many access control models that implement DAC and / or MAC. We now briefly describe three models that are used most often, and at the same time point out their deficiencies in meeting today's web service security needs (Yuan & Tong, 2005).

ROLE-BASED ACCESS CONTROL

Role-Based Access Control (RBAC) is an evolution in terms of Access Control for firms outside the government, whose security needs were not fulfilled by MAC or DAC. Being a more sophisticated Access Control method, RBAC allows for a simplified many-to-many relationship description between individuals and rights (Ferraiolo & Kunh, 1992).

A role was defined as a "Semantic construct forming the basis to access control policy" (Sandhu & Coyne, 1996), a role is therefore a means to grant Access by creating task specific parameters, such as the position held by a person in an organization.

The RBAC model restricts access to resources in an IS based on the function, or role (ergo, the name) that a user performs for a given firm. Coyne, et al. (2013) explained that, in this type of Access Control method, users are

grouped according to their role in the organization, and the permissions are assigned to the roles, rather than assigning them to the users. This inherent property of the system allows it to be easily scaled, without incurring in increased administration costs (Yuan & Tong, 2005). Coyne, et al. also stated that for the AC method to be effective, all the accesses to the IS "must occur through the RBAC system".

Once called "the most attractive solution for providing security features in multidomain digital government infrastructure" (Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001). Also stated by Ferraiolo, et al., "the RBAC model Security administration is also greatly simplified by the use of roles to organize access privileges", as there are usually fewer roles than subjects in the organization, roles are easier to maintain and if a subject changes its role, the task of reviewing and changing its permissions translates into adding it to the new role and eliminating it from the latter, instead of individually granting and revoking its permissions (as would have been the case in older models).

The RBAC model deals with other administrative constraints, which are reviewed as follows: Core RBAC, Hierarchical RBAC, Static Separation of Duty and Dynamic Separation of Duty.

CORE RBAC

As explained above, the basic concept of RBAC is that users are assigned to roles, permissions are assigned to roles, and users acquire permissions by being members of roles (Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001). Also explained by Ferraiolo, et al. (2001) is the fact that RBAC fits into environments where one subject is allowed to belong to various Roles, while many users can belong to a single role. The model also allows a similar rule for permissions, where a single permission can be assigned to several roles, and a role can be assigned to several permissions.

Included in the basic model of RBAC comes an administrative function, named by Ferraiolo, et al. (2001), the user-role review, "whereby the roles assigned to a specific user can be determined as well as users assigned to a specific role. A similar requirement for permission-role review is imposed as an advanced review function". The model also includes the concept of user sessions, this concept allows the activation of a role of a user in a "session", this also helps deactivate and regulate the roles (Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001).

Hierarchical RBAC

Along with the explanation given by Ferraiolo, et al. (2001) of the basic RBAC model (or Core RBAC), this variant of the RBAC model supports, as its name suggests, Role hierarchies. A hierarchy is defined as "a system in which people or things are placed in a series of levels with different importance or status". (Merriam-Webster Inc., 2014)The RBAC model recognizes two types of role hierarchies:

> General Hierarchical RBAC. This type of hierarchy supports the set of an arbitrary partial order to serve as the Role hierarchy, this is meant to include the concepts of permissions inheritance (both from juniors to seniors and vice-versa), and the user membership among roles (Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001).
>
> Limited Hierarchical RBAC. Although some systems may benefit from the implementation of Hierarchy in the RBAC System, these benefits may need some restrictions, such as trees of inverted trees (Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001).

Static Separation of Duty

Administrating any RBAC system, although easier than previous ISs, is not free of conflict among roles themselves. Conflicts arise as a subject gains "permissions associated with conflicting roles" (Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001).

One method to resolve the conflicts is to use Static Separation of Duties (SSD), this is, to create and enforce rules when adding users to the roles. Ferraiolo, et al. (2001) said, "These rules and policies can be centrally specified and uniformly imposed on specific roles". To avoid potential inconsistencies related to SSD, and hierarchical RBAC, when adding the possibility to inherit roles, the requirements of SSD were defined as follows:

- Static Separation of Duty. The relations place constraints on the assignment of users to roles, this may constrain any Subject to belong to one or more Roles, if (according to the enforced policy) the current Role of the subject is in direct conflict with another role (or group of Roles) (Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001).
- Static Separation of Duty in Presence of a Hierarchy. In this type of SSD, the permissions inherited and the permissions assigned to the roles are considered, instead of just considering the assigned permissions.

As an example, think of a Subject whose role in a company is that of Salesman. The company has a constrain regarding the discount that any member of the sales staff can give to a client, sais rule states that each discount must be first disclosed with the marketing staff, before it can be approved. To avoid conflicts, the IS Security administrator in the firm created a rule where none of the members of the Sales-related roles can belong to the Marketing-related roles.

In the previous example, we introduced the idea of a set of roles (Sales and Marketing), and a member of either of the roles can only pertain to a single role in the set. As explained by Ferraiolo, et al. (2001), SSD can be defined as a pair (*role_set, n*) where *role_set* can be a set of conflicting roles and *n* the maximum number of roles that a single Subject may pertain to (*n=1* for the previous example) Figure 3 explains the concept of RBAC, regarding hierarchies and SSD.

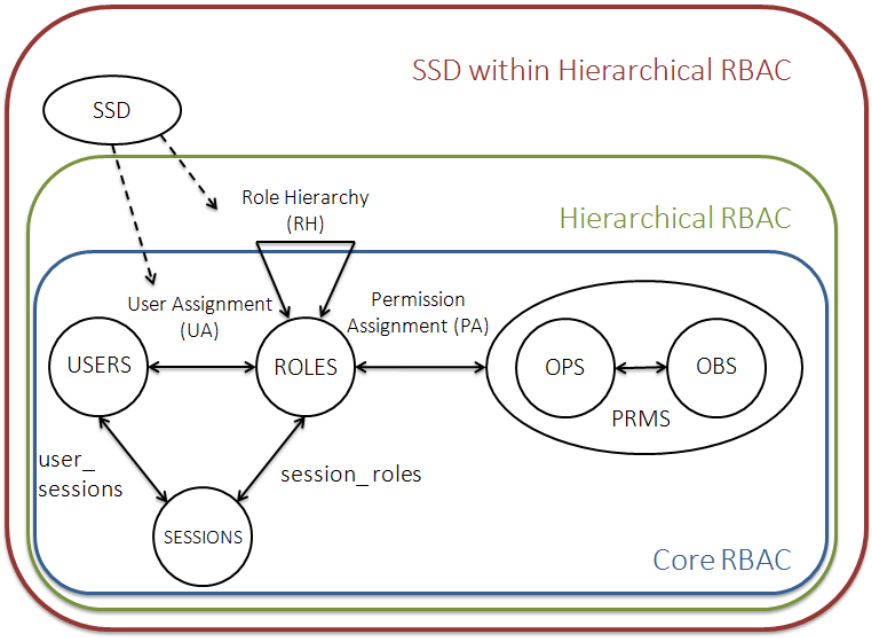

Figure 3. RBAC Concept

Dynamic Separation of Duty

Dynamic Separation of Duty (DSD), differ from SSD as the limitations imposed to a Subject in the pair (*role_set, n*) are defined with $n \geq 2$, with the property that no Subject may activate *n* or more roles from the set of roles (Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001). To see where DSD plays a key part in the RBAC model, please refer to Figure 3.

ATTRIBUTE-BASED ACCESS CONTROL

As its name implies, Attribute Based Access Control (ABAC), uses the attributes of a subject, attributes assigned to the object, other external attributes (often called environmental conditions), to determine the set of policies, and possible actions available for a Subject to exert in an Object (Hu, Ferraiolo, Kuhn, Schnitzer, Miller, & Scarfone, 2014).

An Attribute is defined as a property associated with any entity in the IS, expressed as a name:value pair (Jin, Krishnan, & Sandhu, 2012). If we consider the use of Identities as the key attribute identifier in an ACL, or the use of Roles, as action determinants in RBAC, with the definition provided we can see that Identity and Role can be stated as attributes, instead of action determinants (Hu, Ferraiolo, Kuhn, Schnitzer, Miller, & Scarfone, 2014). This means that ABAC can be thought of as the next logical step in the realm of AC methods.

Hu, et al. (2014) go beyond these statements and add that "The key difference with ABAC is the concept of policies that express a complex Boolean rule set that can evaluate many different attributes"; furthermore, the authors clarify that "While it is possible to achieve ABAC objectives using ACLs or RBAC, demonstrating AC requirements compliance is difficult and costly due to the level of abstraction required between the AC requirements and the ACL or RBAC model". One problem devised by Hu, et al. (2014) is that changing an AC requirement in an RBAC, or ACL model adapted to work as an ABAC model; difficulties can appear while identifying the places where the implementations need to be updated.

When addressing the external attributes, it is noteworthy to review the fact that they may have a larger part in keeping the Objects safe. These attributes are independent of the Subjects or the Objects, but can, nevertheless be taken into account in the AC policies to ensure better

protection. These attributes can take the form of an hour, a date, a location, etc. (Coyne & Weil, 2013), and as they maintain the same name:value pair structure defined above, the attribute framework can be easily applied to them (Jin, Krishnan, & Sandhu, 2012).

It should be recognized that ABAC, with its flexibility might further worsen the problem of role design and engineering, attribute engineering is likely to be a more complex activity but it is a necessary activity to have an increased flexibility (Jin, Krishnan, & Sandhu, 2012). Coyne & Weil (2013) stated that:

> With ABAC, the need to engineer roles is eradicated, as long as role names aren't used as attributes. However, a potentially large number of attributes must be understood and managed, and attributes must be selected by expert personnel. Furthermore, attributes have no meaning until they're associated with a user, object, or relation, and it's not practical to audit which users have access to a given permission and what permissions have been granted to a given user.

Once adopted, as a subject requests access, the ABAC engine can make an access control decision based on the assigned attributes of the requester, the assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions. Under this arrangement policies can be created and managed without direct reference to potentially numerous users and objects, and users and objects can be provisioned without reference to policy (Hu, Ferraiolo, Kuhn, Schnitzer, Miller, & Scarfone, 2014).

Coyne and Weil (2013), stated on their report the following:

> Conceptually, ABAC and RBAC are similar. It is the properties of each model that give them their nature and behavior. […] Combining ABAC and RBAC isn't an architectural challenge. Each model would have its

own rule base in the policy information point (PIP). The policy decision point (PDP) would need the capability to evaluate these rules to produce an access decision.

Thus, it's possible to obtain the flexibility and advantages of ABAC while maintaining RBAC's advantages for analysis and risk control, if roles are used to define the maximum set of permissions that users can have.

DIGITAL RIGHTS MANAGEMENT

Allowing content users to define, and enforce restrictions on how the content is used (Ma, Huang, Yang, & Niu, 2013), Digital Rights Management (DRM) provides a secure and reliable way to distribute contents online. The restrictions currently allowed by DRM include "Publishing architecture, business models for online content distribution, digital policy management, privacy and anonymity, security including encryption, authentication and authorization, tamper resistance, and watermarking, traitor tracing, broadcast encryption, obfuscation, usability aspects of DRM systems" (Ma, Huang, Yang, & Niu, 2013).

The importance of Authorization in DRM is also explained by Ma, et al. (2013) as follows:

> In a DRM system, authorization is one of most important issues should be taken into account, for the whole procedure of DRM relies on the authorization to get rights of the protected content, which includes authentication of principal, license creation, releasing, revocation and transferring, during the whole process it must ensure data security, integrity, and fairness and non-repudiation of the transaction, and it must maintain the authorization for latent update of license for rights management.

The idea behind DRM is to create a Security Reference Monitor from the side of the client. A reference monitor "observes software execution and takes remedial action on operations that violate a policy" (Crampton, 2005). This approach differs from traditional AC methods, as these are based entirely upon the side of the server.

Park & Sandhu (2004) explain, "Usage decisions in commercial DRM solutions usually utilize user-defined, application-level, payment-based security policies, and do not include traditional access control policies. Typical examples include pay-per-view, metered payment, membership-based monthly subscriptions, and so on".

One clearly stated setback of the DRM model is that "DRM has hardly recognized commercial B2B transactions in their solutions though its underlying technologies can be used for controls on this kind of sensitive information usage" (Park & Sandhu, 2004). For most users of online electronic content stores, such as iTunes Store, or Kindle Store, it is evident that the seller gives clear indications on the limitations of the customer, as they are purchasing the rights to reproduce the information inside the files. But, as stated by Park & Sandhu (2004), these benefits have been mostly exploited in commercial B2C transactions, as the ownership of the information does not go through middlemen.

USAGE CONTROL

Before giving a working, and ample definition of Usage Control (UCON), it is important to state that, along the lines used to define the succession of models presented above, "UCON is not a substitute for traditional access control, trust management, or DRM." (Park & Sandhu, 2004). Moreover, UCON has been proposed as a complete, thorough, and expanded solution regarding the three areas mentioned before. A key component in UCON is the fact that even after the dissemination of the content (a change of

ownership in the information), the model still achieves "fine-grained control on digital resources" (Park & Sandhu, 2004).

Park & Sandhu (2004), go beyond the mere idea of creating a more thorough method to control the Access, and the Usage of information, and state, "The goal of UCON is to provide a new intellectual foundation for access control." The idea is furthermore expanded by adding, "The 30-year-old framework of the access matrix has been extended in various different directions as researchers have found it to be inadequate for their needs." (Park & Sandhu, 2004). These needs, as well as the common points that the model has with ABAC are explained by Park & Sandhu (2004):

> Traditionally, access control has dealt with authorizations as the basis for its decision-making process. In the UCON model, the authorization-based decision process utilizes subject attributes and object attributes. Attributes can be identities, security labels, properties, capabilities, and so on. The UCON model includes obligations and conditions as well as authorizations as part of usage decision process to provide a richer and finer decision capability. The necessity of obligations and conditions has been recognized in modern business systems such as B2C mass distribution systems as well as B2B transactions and interactions between business partners. Obligations are requirements that have to be fulfilled for usage allowance. Conditions are environmental or system requirements that are independent from individual subjects and objects.

Moreover, just as RBAC, and ABAC, UCON is a session-based AC model, as Stated by Zhang, et al. (2005) "it controls the current access request and ongoing access".

MUTABILITY AND CONTINUITY: THE UCON CORE MODEL

After what meant ABAC as a conceptual advance in AC methods, it may be hard to imagine how Park & Sandhu may try to differentiate UCON, whilst going above and beyond the scope of traditional AC methods. These properties of UCON are "the continuity of access decisions and the mutability of subject and object attributes" (Zhang, Parisi-Pressice, Sandhu, & Park, 2005).

Continuity in UCON terms means that "authorization decisions are not only checked and made before an access, but may be repeatedly checked during the access and may revoke the access if some policies are not satisfied, according to the changes of the subject or object attributes, or environmental conditions" (Zhang, Parisi-Pressice, Sandhu, & Park, 2005). The IS administrator must have special care when defining the AC policies, as the attributes of the Subjects and the Objects may be changed during the course of an active session, as these change, the permissions held must be revised before, during, and after the session.

Mutability, as a concept is newly introduced by UCON, but as explained by Zhang, et al. (2005) "its features can be found in traditional access control models and policies."

Zhang, et al. (2005), explain the concept as follows:

> For example, in a Chinese Wall policy, if a subject accesses an object in a conflict-of-interest set, then he/she cannot access any other conflicting objects in the future. That means, the potential object list that the subject can access (we can consider this a subject attribute) has been changed as a side effect of a previous access. This change, consequently, will restrict the next access of this subject. History-based access control policies can be expressed by UCON with this

> feature of attribute mutability. Mutability is also useful to specify dynamic constraints in access control systems, such as separation-of-duty (SoD) policies, cardinality constraints, etc.

Park & Sandhu (2004), make the following statement about the decision-making process:

> One of the basic assumptions in the $UCON_{ABC}$ model is that its decision-making process is transaction-based. This means that decision predicates are evaluated upon each usage request and the decision influences usages of that request. A *pre* decision predicate decides approval or denial of the request. An *ongoing* predicate may revoke or continue to allow current exercise of the requested usage.

Figure 4 shows the definition if Continuity and Mutability when applied to authorization decisions, policies, and attributes. The actions and components necessary to check and enforce Access decisions (i.e. Continuity) lie in the first two phases, the pre-decision, and ongoing-decision phases, as there is no Access to resources in the after usage phase, there is no need to check or enforce any type of decision in it (Zhang, Parisi-Pressice, Sandhu, & Park, 2005); however, Conditions and Obligations can be defined in this last stage (the complete definition of what are Conditions and Obligation is found below).

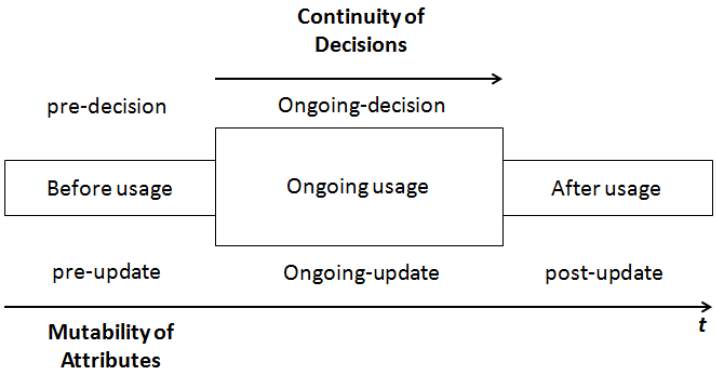

Figure 4. Continuity and Mutability in UCON

Obligations and Conditions can be considered as long-term Obligations and Conditions, as long as these are held during the after-usage phase. These long term Obligations and Conditions do not fall into the scope of Core UCON, but Zhang, et al. (2005) encourage including them in the administrative models of the AC method.

Mutability, on the other hand, means that attributes defining a subject or an object attributes can be updated as a direct result of an access (Zhang, Parisi-Pressice, Sandhu, & Park, 2005). Along with the three phases defined in Figure 4. Continuity and Mutability in UCON, there are three types of updates related to each one of them (pre-updates, ongoing-updates, and post-updates). An update of an attribute of a subject or an object during a session may, as a consequence, result in a system action allowing or revoking the current access or another access, according to the authorizations of the access (Zhang, Parisi-Pressice, Sandhu, & Park, 2005). Also defined by Zhang, et al. (2005), "an update on the current usage may generate cascading updates, while an update on another access can act as an external event that would cause a change of the usage status, such as revocation".

Zhang, et al. (2005) explain a differentiating feature of UCON, when compared to RBAC and ABAC:

> Unlike administrator-controlled, in system-controlled attribute management, updates are the side effects or results of a subject's usage on objects […]. This is different from the update by an administrative action because the update in this case is done by the system, while in administrator-controlled management the update involves administrative decisions and actions. This is why system-controlled attributes are "mutable" attributes that do not require any administrative action for updates.

Another difference between traditional Access Control models is the focus they have on Server-side control only, this left Client-side control aside, and gave an ability to control the usage of digital resources after these were distributed. Explained by Park & Sandhu (2004) "The UCON model can be utilized in both server-side and client-side control architectures though some functional details are likely to be different."

THE UCON$_{ABC}$ CORE MODEL

The UCON$_{ABC}$ model consists of eight core components. They are subjects, subject attributes, objects, object attributes, rights, authorizations, obligations, and conditions. Authorizations (A), oBligations (B), and Conditions (C) are common ground rules that have to be evaluated for usage decision.

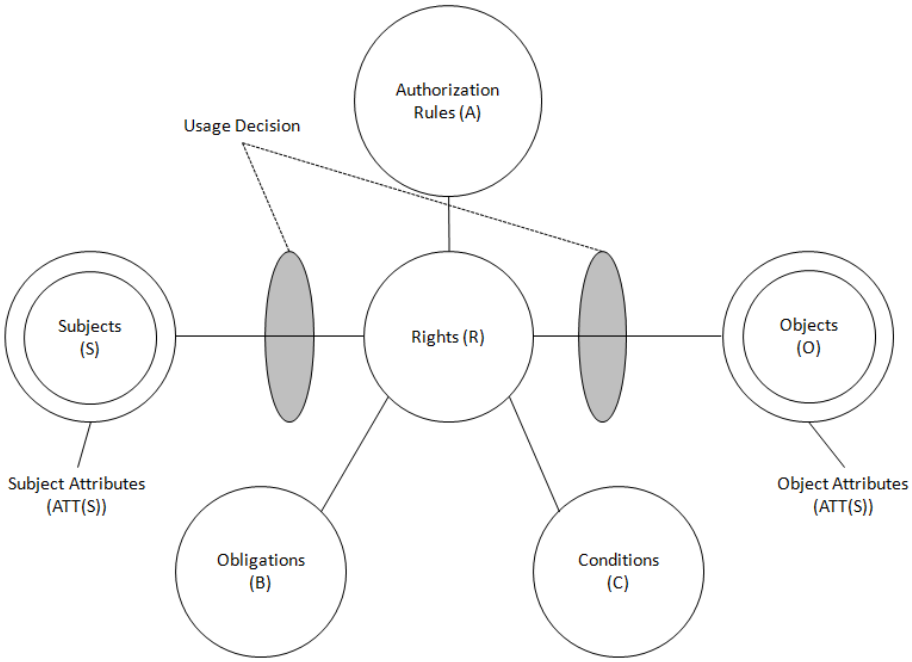

Figure 5. A graphical representation of UCON components

SUBJECTS AND SUBJECT ATTRIBUTES IN UCON$_{ABC}$

As is the case with UCON, going beyond the scope of previous AC models, in UCON, Objects have relationships with three types of subjects: Consumer, Provider and Identifiee Subjects (Park & Sandhu, 2004), instead of just Consumer, and Provider, which is the case for BAC and RBAC. The motivation to include the Identifiee Subject in the mix, is to incorporate Privacy protection, as well as Intellectual Rights Property Protection (Manly a DRM issue), and Sensitive Protection Information (an AC issue), in an AC model that has both a Client-side Reference Monitor, and a Server-side Reference Monitor (See Figure 6. Area Coverage of different AC methods)

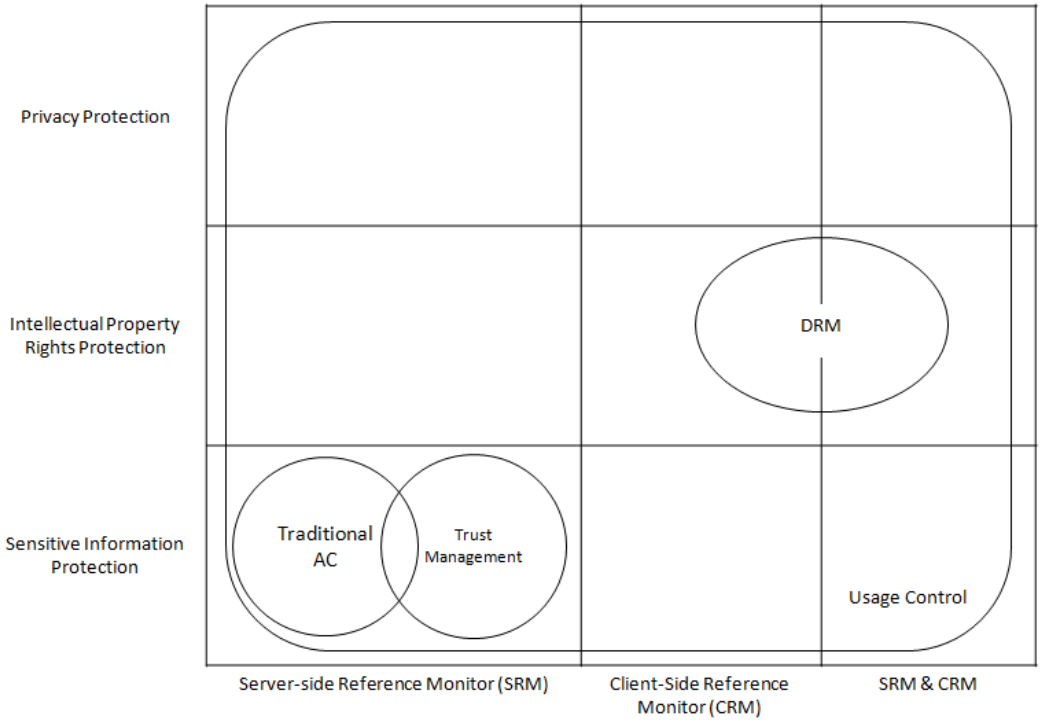

Figure 6. Area Coverage of different AC methods

The following descriptions for each subject are interpretations of those given by Park & Sandhu (2005):

- Consumer subjects (CS) receive rights (and in some cases objects) and use them to access objects (whether held in the IS or given to the Subject).
- Provider subjects (PS) are entities whose role is to provide objects and, depending on the Administrative Policies, hold certain rights on it.

- Identifiee subjects (IdS) are entities who are identified in digital objects that include their privacy-sensitive information.

Furthermore, Park & Sandhu (2005) give the following clarification regarding IdS: "Although the concept of identifiee subjects always exists in case of

privacy-sensitive information, identifiee subjects may or may not be included within UCON systems based on other control policies."

Regarding the attributes of the Subjects, they comprise those of ABAC; however, they are not limited to them, the $UCON_{ABC}$ model includes the definition of mutable attributes, and immutable attributes. Park & Sandhu (2004) state that: "If an attribute is immutable, it cannot be changed by the user's activity. Only administrative actions can change such an attribute. A mutable attribute can be modified as a side effect of subjects' access to objects."

OBJECTS AND OBJECT ATTRIBUTES IN $UCON_{ABC}$

In $UCON_{ABC}$, Objects are associated with attributes; the attributes can be associated by themselves or with Rights. Objects in UCON have the property of being either privacy-sensitive, or not. Privacy-sensitive Objects include individually identifiable information that can cause privacy problems if not managed properly. (Park & Sandhu, 2002).

An UCON object can be either original or derivative. A derivative object in UCON is different from that of other DRM literature, as derivative Objects in DRM are cited, quoted or copied Objects that include work from an original source (Park & Sandhu, 2004). In UCON, a derivative Object is an Object created as a consequence of obtaining or exercising rights on an original object. Just like the original Object, derivate Objects must hold relations with other components of the AC method, as well as maintain the proper UCON properties in order to provide protection on the rights of all involved subjects (consumer, provider, and/or identifiee subjects) (Park & Sandhu, 2004).

Objects, as well as subjects, are divided among the same three kinds: Consumer, Provider, and Identifiee.

RIGHTS

Park and Sandhu (2004) briefly state the meaning of Rights in the $UCON_{ABC}$ model as follows: "Rights are privileges that a subject can hold and exercise on an object". This concept is similar to a Right in other AC methods, and as such, they can be hierarchical, or not. Like Subjects, and Objects, Rights can also be divided into the same three categories (Consumer Rights, Provider Rights, and Identifiee Rights), to provide a fine-grained control over the Subjects. From an AC point-of-view, having the appropriate rights enables a Subject to exert them on an Object in a specific way (such as read, modify or delete) (Park & Sandhu, 2004).

Although the Rights may be similar, there is a difference from the $UCON_{ABC}$ point-of-view, this type of AC does not represent, or visualize a right as a field in a Matrix, whether the Subject uses it or not. The existence of the right is determined when the Subject attempts the access to an Object. In general, rights are divided into Direct Usage Rights (such as read), delegation of rights (In the case of hierarchies, DSD, or SSD), and rights for administering access (such as modify subject and object attributes that in turn determine access rights) (Park & Sandhu, 2004).

AUTHORIZATION RULES

Authorization rules, defined by Park & Sandhu (2002) are "a set of requirements that should be satisfied before allowing subjects' access to objects or use of objects". Authorizations evaluate Subject attributes, Object attributes, and requested Rights together with authorization rules for usage decision and are divided in two types, Rights-related Authorization Rules (RAR), and Obligation-related Authorization Rules (OAR). RAR are used to check if a given Subject has the necessary privileges to exercise certain rights on an Object. OAR exist to check if a subject has agreed to fulfill an oBligation on an object, after the Subject has obtained or enforced the Rights it has on an Object (Park & Sandhu, 2002).

Authorizations are no different than the rest of Attibutes that compose the $UCON_{ABC}$ model, as they can mutate before and during the usage of an Object. As such, authorizations can be either pre-authorizations (preA) or ongoing-authorizations (onA) (Park & Sandhu, 2004). preAs are performed before a requested Right is exercised, while onAs are performed while the right is exercised, onAs may be performed continuously or periodically during the time span of access (Park & Sandhu, 2004). Traditional AC policies (MAC, DAC, RBAC, ABAC and TM) use some form of pre-authorization for their decisions. An example of ongoing-authorization would be the continued checking of revocation status during the exercise of usage. Thereby, usage can be immediately terminated to enforce immediate revocation (Park & Sandhu, 2002).

Authorization rules are different from Conditions, as "Authorization rules are a set of decision factors used to check whether a subject is qualified for the use of certain rights on an object, whereas the condition is used to check whether existing limitations and status of usage rights on an object are valid and whether those limitations have to be updated" (Park & Sandhu, 2002).

oBLIGATIONS

Obligations are "functional predicates that verify mandatory requirements a subject has to perform before or during a usage exercise" (Park & Sandhu, 2004).

oBigations, just as Authorizations, can be pre-obligations (*preB*) or ongoing-obligations (*onB*). *preBs* are predicates that use some kind of history functions to check if certain activities have been satisfied or not and returns a Boolean argument (either 'true' or 'false'). *onBs* are predicates that have to be satisfied continuously or periodically while the allowed rights are in use (Park & Sandhu, 2004).

Regarding the use of attributes by oBligations, Park & Sandhu (2004) state:

> Obligations may or may not use Subject or Object attributes. Attributes can be used to determine what kind of Obligations are required for usage approval. Obligations may require certain updates on subject attributes. These updates are likely to affect either current or future usage decisions. Note that attributes are not used for decision making with respect to obligations, but only for choosing what obligations apply.

As seen at the beginning of this chapter, the decision-making process in $UCON_{ABC}$ is transaction-based; however, in contrast with Authorizations or Conditions, there can be global oBligations independent of any transaction, these global obligations exist where the oBligations influence only upcoming time-based or event-based)" (Park & Sandhu, 2004).

CONDITIONS

Conditions are "a set of decision factors that the system should verify at authorization process along with authorization rules before allowing usage of rights on a digital object" (Park & Sandhu, 2002). These can belong to one of two types, Dynamic Conditions or Static Conditions.

Dynamic Conditions include information that should be checked for updates each time the Subject request Access (or Usage) of an Object. Static Conditions include information that does not necessarily have to be checked for updates. Moreover "Dynamic Conditions are stateful and the Static Conditions are stateless" (Park & Sandhu, 2002)

Conditions evaluate current environmental or system status, checking if relevant Usage requirements are satisfied or not and return a Boolean argument.

The usage of attributes as Conditions is discussed by Park & Sandhu (2004):

> Subject attributes or object attributes can be used to select which condition requirements have to be used for a request. However, no attribute is included within the requirements themselves. Unlike authorizations or obligations, condition variables cannot be mutable, since conditions are not under direct control of individual subjects. Evaluation of conditions cannot update any subject or object attributes.

The authors of the $UCON_{ABC}$ model address the usage of devices and add them to Conditions, arguing that this is preferred "since there are other subject and object independent factors such as time periods and system load" (Park & Sandhu, 2004).

Conditions differ from authorizations as conditions "mainly focus on evaluations of environmental, system-related restrictions that have no direct relationship with subject and object attributes for usage decision (that is, subject and object attributes are not included within condition requirements hence not required for usage decision process)" (Park & Sandhu, 2004), while authorizations check attributes related to Subjects and Objects.

METHODOLOGY

The methodology that will be used for the rest of the project was developed by Gouglidis and Mavridis (2009); in it, Access Control methods are compared based on four layers: Logic, Management, Assets and Entropy. Although the work of Gouglidis and Mavridis was meant to assess the performance of RBAC, ABAC, and UCON in Grid Computing, it can be extended to other Computing realms as long as the layers are properly specified.

ENTROPY LAYER

In this layer, the characteristics of a system, and its distribution are revised. The entropy "refers to the virtual and geographic spatial distribution of a system and the fluctuations of sharable resources in time" (Gouglidis & Mavridis, 2009). The characteristics of the system refer to the resources, homogeneous and heterogeneous that make it. The definition of a resource used throughout the document is different to the one used here, as the resources of the system are not only software objects, but also hardware objects.

In this layer, the resources of the system are coupled with their spatial distribution, as well as their changes through time. This extends the variety of systems that can be described.

ASSETS LAYER

Gouglidis and Mavridis define an asset as "a resource in a grid system that can be shared within a Virtual Organization". In this layer, all the assets that make part of an IS are taken into account. Assets are classified in two classes: software or hardware.

The software class is divided in two subclasses: service, and data. The hardware class is divided in three subclasses: Computational (i.e. the hardware characteristics of the system such as CPU frequency, the size of the RAM, etc.), Storage (storage space), and Equipment (I/O devices).

MANAGEMENT LAYER

The Management Layer captures and translates the security needs raised from the management policies within the firm.

The distribution of a system determines the policy management, which can be centralized or decentralized. The authors of the comparison and assessment methodology mention that "systems with a low level of spatial distribution, as defined in the entropy layer, require a centralized management subsystem and vice-versa" and that "a small local business application using grid technologies requires centralized management" (Gouglidis & Mavridis, 2009).

The enforcement policies discussed above must be taken into account as well, and as such, the authors classify them in static or dynamic enforcements.

The level of intervention of an administrator in managing the routines, or level of automation must be addressed. Fully automated systems are those in which the management is done by the system itself, Semi-automated systems are partially managed by themselves and by administrators, and finally automation-absent systems are entirely administered by humans. In this layer Problem identification, conflict resolution, and revocation of privileges are considered.

LOGIC LAYER

The logic layer is where application models and types of execution are located. It is divided into two classes; the first class pertains to the abstract class of models, were the security issues that rise from the applications being executed in the system are addressed. The second class is that of the nature of the business (e.g. Business, Science, Governmental, etc.).

The authors propose another type of classification based on the requirements of the applications. In it, applications are divided into batch-based execution, and interactive-based execution:

> Usually, science projects require a batch-based execution of applications in order to provide results through the computation of data. In contrast, most business applications demand an interactive environment to tackle the highly dynamic enterprise environment.

With the layering, and classification explained, Gouglidis and Mavridis apply the aforementioned concepts to the three types of Access Control addressed in this project: RBAC, ABAC, and UCON.

ACCESS CONTROL APPLICATION

As for the application of the four layers on each of the three AC models, the authors arrive to the following conclusions.

ENTROPY LAYER

As ABAC and UCON natively support attributes, the environment distribution is not an obstacle for the two AC methods. Gouglidis and Mavridis (2009) state that "Attribute repositories can be dispersed across different domains and attributes can easily be retrieved. The use of attributes also overcomes the heterogeneity problems in VOs."

The flexibility of the two models allow them to deal with dynamic modifications in the entities present in the IS, whilst RBAC handles centralized architectures in a more suitable way, as participants are defined before the model is setup.

Assets Layer

As happened with environment distribution in the entropy layer, ABAC and UCON are fine-grained access models, this means that the solutions for the security needs for each entity in the IS are as detailed as required, and thus, very flexible. Core RBAC is a coarse-grained AC model, as the subjects or objects have to be grouped under roles; unlike ABAC and UCON, in order to decrease the grain size in core RBAC, a new role must be created, this increases the complexity of the model, making the model less flexible to changes.

Management Layer

The lack of flexibility in RBAC increases the administrative capabilities in the AC model, as assignments are build in a user-to-role and role-to-permission mode. As seen above in the Bibliographic Review, the strengths of RBAC rely on the ease to revoke user assignments, create role hierarchies, as well as delegation constraints. But, as said above distributed systems, hence distributed management is not a strong point of RBAC; ABAC and UCON handle better distributed management issues. Unlike ABAC, the mutability attributes in UCON allow to take immediate actions regarding the usage of resources as well as changes in policies; although UCON is more prepared to what may seem as a wider selection of scenarios, Gouglidis and Mavridis (2009) explain that "Conflict resolution can be cumbersome in both [ABAC and UCON]", and that "Automation is absent from all models".

LOGIC LAYER

While ABAC and UCON seem to excel in the Entropy and Assets layers, and RBAC does the same in the Management layer, Gouglidis and Mavridis (2009) explain that requirements in the logic layer are not handled properly in the examined models, but that UCON supports interactive environments due to the continuity of decisions and mutable attributes. Regarding resources and requirements the authors state, "Limitations on resources can be enforced through conditions and obligations can handle a number of business requirements. Still, topics like service composition are left intact."

AC Models	**Conceptual Categorization Layers**			
	Entropy	**Assets**	**Management**	**Logic**
ABAC	High	Medium	Low/Medium	Low
RBAC	Low/Medium	Low/Medium	Medium/High	Low
UCON	High	Medium	Medium	Medium/High

Table 1. Evaluation of AC models (Gouglidis & Mavridis, 2009).

Table 1 gives a brief summary on how each AC model handles each of the proposed layers. This evaluation will be used throughout the following section, in which a scenario using ABAC and UCON is proposed.

RESULTS

To show how do ABAC and UCON may behave in a real world implementation, the paper entitled "The Role-Based Access Control System of a European Bank: A Case Study and Discussion" by Schaad, Moffett, and Jacob (2001) will be used as a reference point. In the aforementioned paper, the authors discuss how did an RBAC implementation aided the bank to better streamline processes within an organization with a large number of user population (Subjects).

This particular case is used as it represents a challenge for both ABAC and UCON, mostly because they allow finer grained control over RBAC. Coyne, et al. (2013), have shown that RBAC may be more suitable for organizations with a high number of subjects, and within a stable environment; while ABAC may be better in smaller organizations, in unstable environments.

The ABAC and UCON implementations were built around the concept of avoiding changes in the interaction between the employees and the IS of the firm, in order to decrease the amount of necessary training that would be required if the interaction changed dramatically.

RBAC IMPLEMENTATION

The implementation of RBAC in the paper written by Schaad, et al. (2001) was set as an answer to an internal issue experienced by the Dresdner Bank. The bank had already created a loosely coupled solution to reduce the overhead staff that the company had hired to maintain all the programs that were necessary for the bank to properly function. As there were no commercial solutions, the Dresdner Bank created its own AC module, the Funktionale Berechtigung System (often abbreviated FUB System).

One of their biggest challenges was that the programs used by the bank came from different backgrounds, in the words of Schaad, Moffett and Jacob (2001): "The Bank uses a variety of different computing applications to support its business, many of which have their origin in the mainframe world, but also more recently deployed client-server based systems." On top of that, the company had over 50,000 employees working at over 1,400 branches around the world, which meant that the new system had to be distributed to each branch, while staying online all the time.

FUB Access Principles

In order to determine whether access should be granted, or not, the system evaluated the access rights for each individual user. The access rights are given to roles created as combinations of pairs of job function and official position within the bank.

To simplify the previously existing AC methods, every application launched by an authenticated user has no knowledge of the permissions and policies the user is constrained with; to do that, the application sends a query to the FUB regarding the security profile of the user, and obtains the necessary information.

Ideally, each employee belongs to one, and only one organizational unit; however, in special cases (such as the absence of a colleague) each user may be given a maximum of four roles.

Regarding the distribution of Access Rights within the firm, the Access Rights concerning any given organizational unit are limited to the employees within the branch. Nonetheless, applications whose access rights are not subject to location constraints can be accessed from every access point in the bank.

The authors proposed the RBAC model as a solution to some issues that were not fully addressed by the FUB system.

ABAC Implementation

It can be assumed that, given the amount of evidence presented in this project, the ABAC model can create a more suitable solution than the RBAC solution. As it is based on attributes instead of roles, the issues regarding External Consultants, Freelancers and Hierarchies can be tailor-made, and modified according to the actual situation of the bank.

Basic System Structure

When compared to FUB, the proposed ABAC implementation may compute roles (which are now an attribute), as a sum of groups, which are a combination of official job position and job function just as the FUB system did with the roles. The bank can now add new information regarding access limitations to each specific application, depending on external attributes (such as date, time, location, etc.).

Subjects (Users or Employees) not only have their Role as a Policy determinant of their Access Rights and Privileges, these can now be determined using their type of affiliation to the company (Freelancer, External Consultant, Intern, fixed-term contract, open-term contract, etc.), and whether or not the Access Rights for that specific role are inherited (i.e. True or False) (See Figure 7).

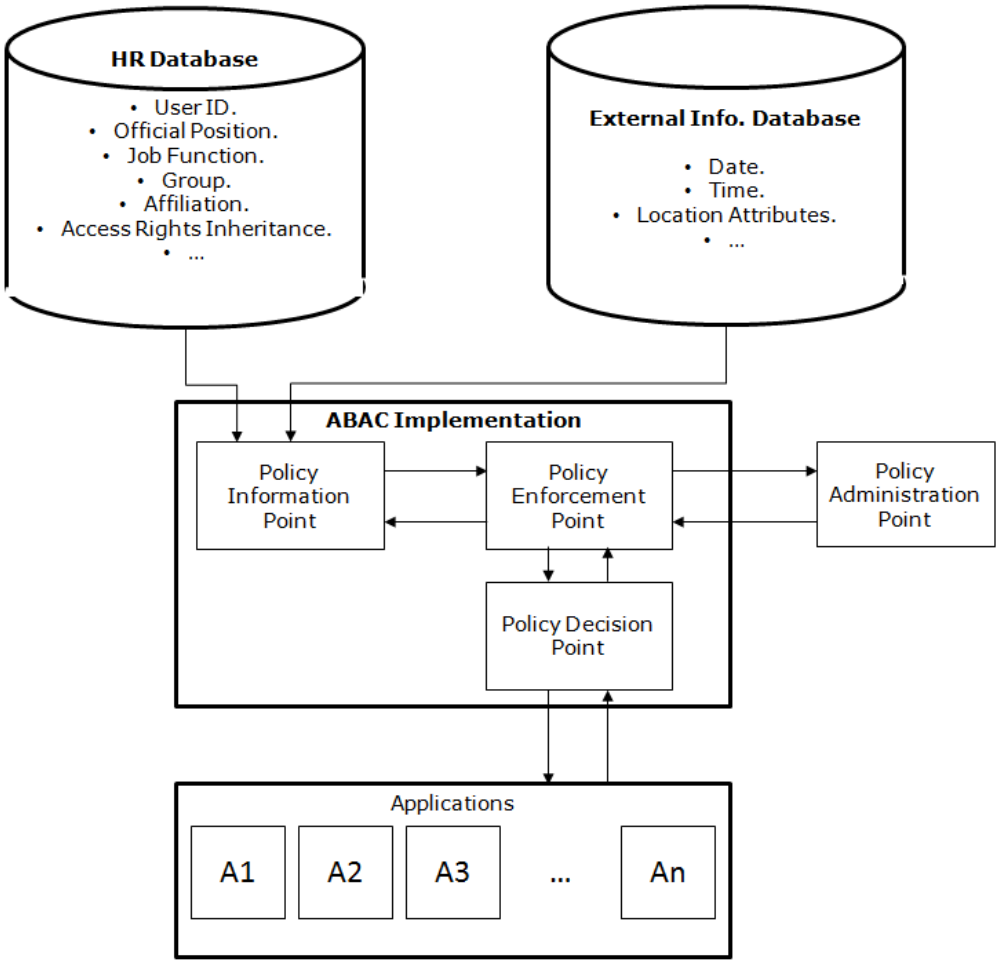

Figure 7. Basic Structure of the ABAC Implementation.

Each Application would still receive the Security Profile for each individual user, as it had whilst using FUB. The Security Profile would now be assigned in a more rigorous way through the use of the mentioned attributes.

As the difficulty to set a Security Profile increases, the need to create a PAP becomes apparent. The PAP should manage the relation between Access Rights and the Functional combinations of User Attributes, the choice of Functional Attributes over Possible Attributes is pointed out by Schaad, Moffett and Jacob (2001) "certain possible roles such as secretary/Member of the Board do not occur in reality" (See Figure 8).

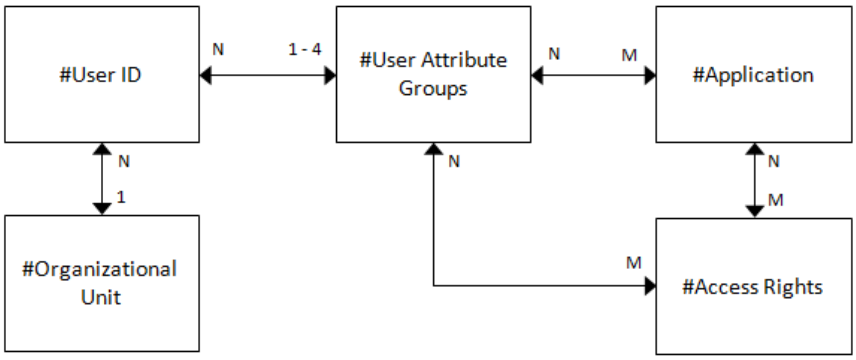

Figure 8. The ABAC Implementation Data Model.

The implementation is intended to manage low-level security clearances locally, with a copy of the Security Profile on each branch. To comply with the security needs of the bank, a background check of every Security Profile stored could be run overnight, in order to update the changes made during the day on the PAP, PDP and PIP.

Access Request

As stated above, the Applications would still request and receive a Security Profile for each Subject. The substantial changes come from the management of the request from the ABAC side, in Figure 9 it can be seen how the ABAC implementation would behave in order to accommodate the changes proposed.

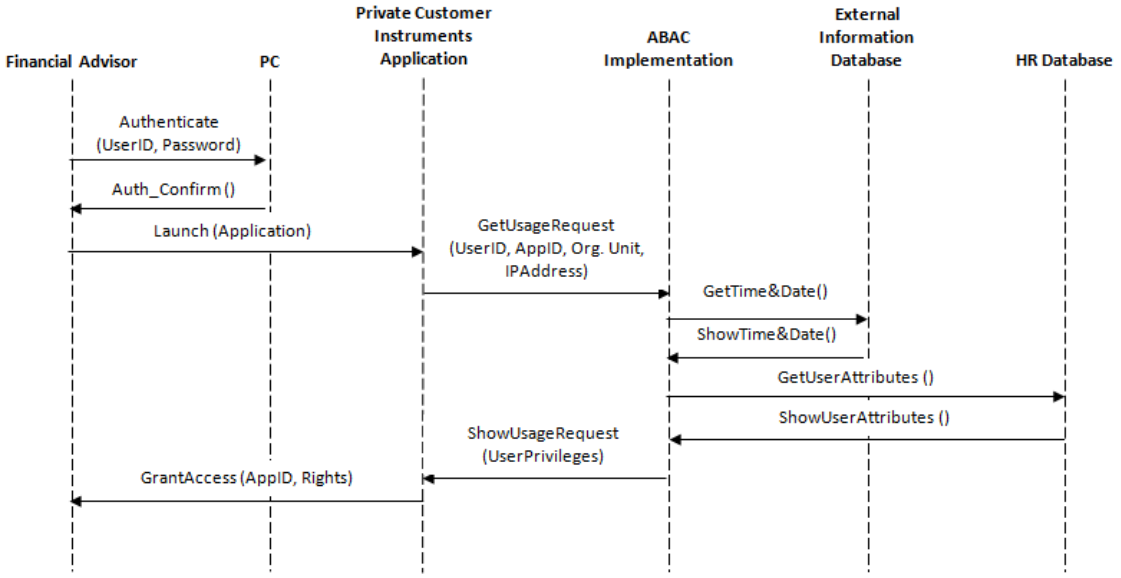

Figure 9. Access Request in the ABAC Implementation.

With the request of the security profile, a new attribute (the IP address of the computer) is sent to the local repository. The local repository then checks the ApplicationID and the local PDP decides if the request can be managed and enforced locally, or not.

If a copy of the Security Profile Requested exists in the local ABAC PIP, access would be granted after the local External Database is checked in order to assess if given the external conditions access should be granted. If, on the other hand, the local distribution could not handle the security profile, a request would be made to the central ABAC PIP, asking for the specific security profile, once there, the Request would be handled in the same way, only that the scale would be larger.

To grant the access, the ABAC implementations (local and central) must check the user attributes needed to grant the Access Rights, and compare them with the profile of the Subject (of which a copy must exist on the Human Resources Database). In that way, if a Subject does not fulfill the

required criteria, the PEP will not allow any operation to be performed on the Objects.

SYSTEM ADMINISTRATION

To maintain the existing Separation of Duties, the HR Department of the bank would still create, modify, or delete the Subjects and their Attributes. The application administrator would still assign the access rights for each individual application, and the ABAC Implementation manager would still do the Subject Attributes/Application assignment.

To avoid breaking the control principles when dealing with the requirements of Freelancers and Consultants, a new position, the Policy Administrator was created (See Figure 10). The goal of the Policy Administrator is to create a link between the Access Rights (used by the Application Administration) and the Subject Attributes (a duty extension taken care by the HR Department), in order to do that, the Policy Administrator should revise the attributes of existing subjects, and use them to decide how to fulfill the security requirements of the bank, while using the principle of Least Privileges to determine the freedom of each Subject in the IS.

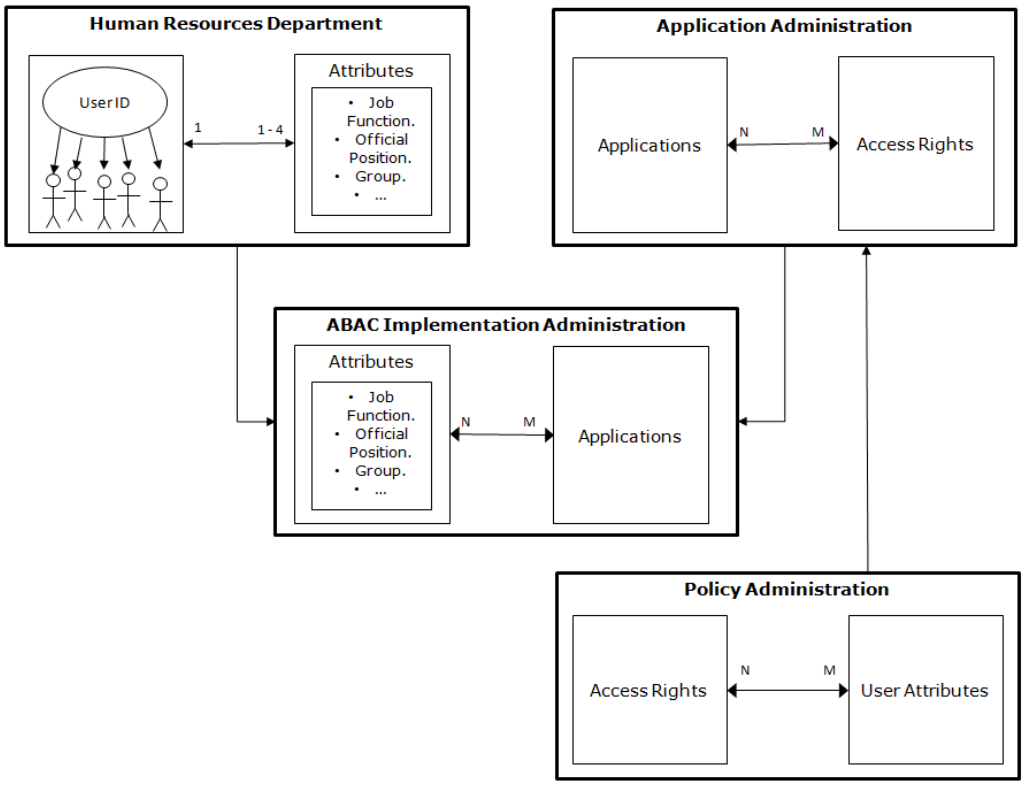

Figure 10. ABAC Administration.

UCON IMPLEMENTATION

While the ABAC implementation may seem to cover the needs of the organization, the mutability found in UCON simplifies most (if not all) of the problems an organization may have when facing mergers, acquisitions and other types of changes in the organization. As UCON relies on attributes to determine which Authorizations, oBligations, and Conditions to apply on top of the rights, in order to use an Object, all the changes previously proposed in the ABAC Implementation would still be enforced.

To ease the explanation of the proposed UCON implementation, as well as reducing the redundancy of it, the UCON implementation will be compared with the ABAC implementation.

BASIC SYSTEM STRUCTURE

The proposed UCON implementation would compute Access Permission as a combination of Attributes that define the Role, Job Function, Official Position, Company Affiliation, and Group just as in the proposed ABAC implementation. Using external attributes, as well as *status* attributes, the AC implementation could change any change in the attributes (for Subjects and Objects) while it is online, decreasing the amount of downtime needed for maintenance, or reducing the need for overhead support of the AC when dealing with sudden changes in the environment or in the subjects.

Subjects would have *status* attributes (e.g. Online, Absent, Dismissed, etc.), which would change the behavior of the AC implementation for them, and for the users directly affected by a change in their status. Rights, Authorizations, oBligations, and Conditions would change depending on the usage from the Subject.

Just as in ABAC, the Applications would receive Security Profiles each time they are used; however, if a change in the attributes occurs before, during, or after the usage, the Security Profile for a given Subject will effectively change between Usages.

To include Authorizations, oBligations, and Conditions, the structure of the AC implementation must be changed in order to include a more robust PAP, able to handle real time changes, as well as adding a Rights, Attributes, oBligations, and Conditions Database, in order to track the history, and add continuity to the changes in the AC Rights (See Figure 11).

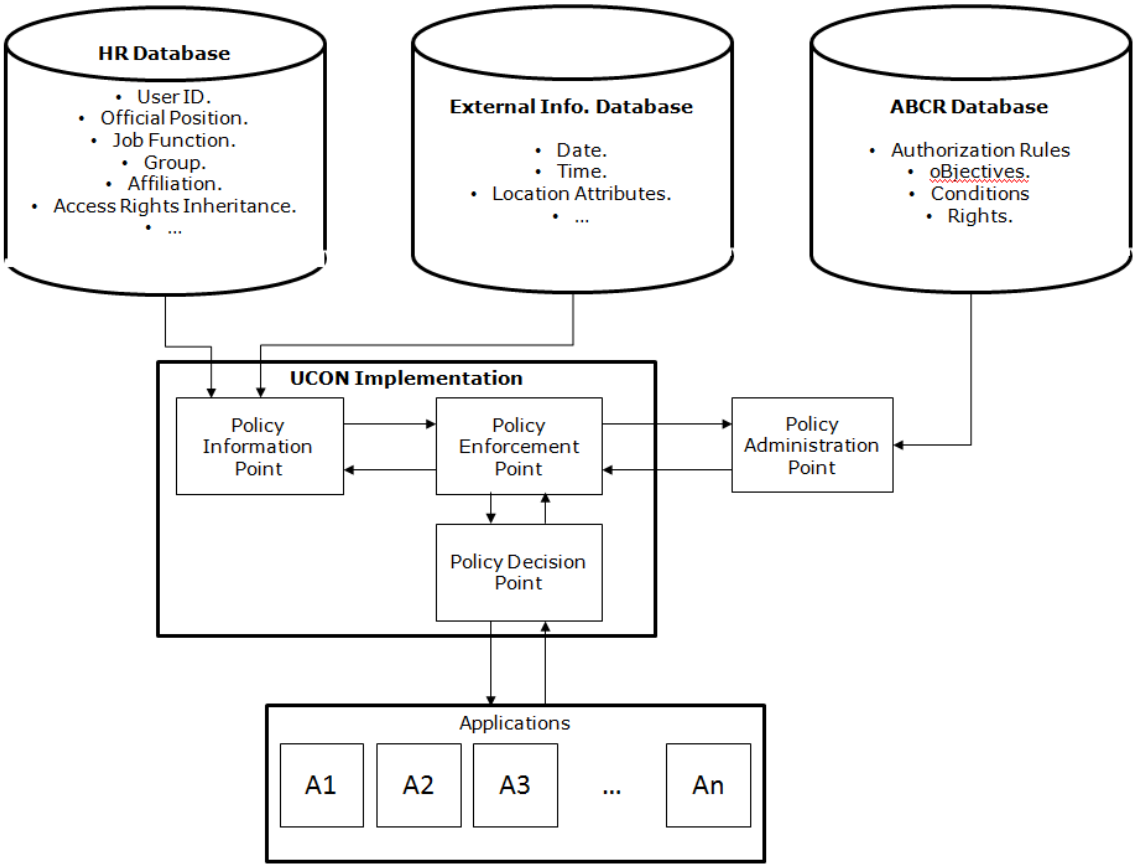

Figure 11. Basic Structure of the UCON Implementation.

The implementation can be thought of being far more flexible regarding tailor-made controls for Subjects to Access Objects, as well as reducing the opportunity for secured information leaks using the same methods more than once (if any information leak happened due to unspecified Attributes or Usage Conditions).

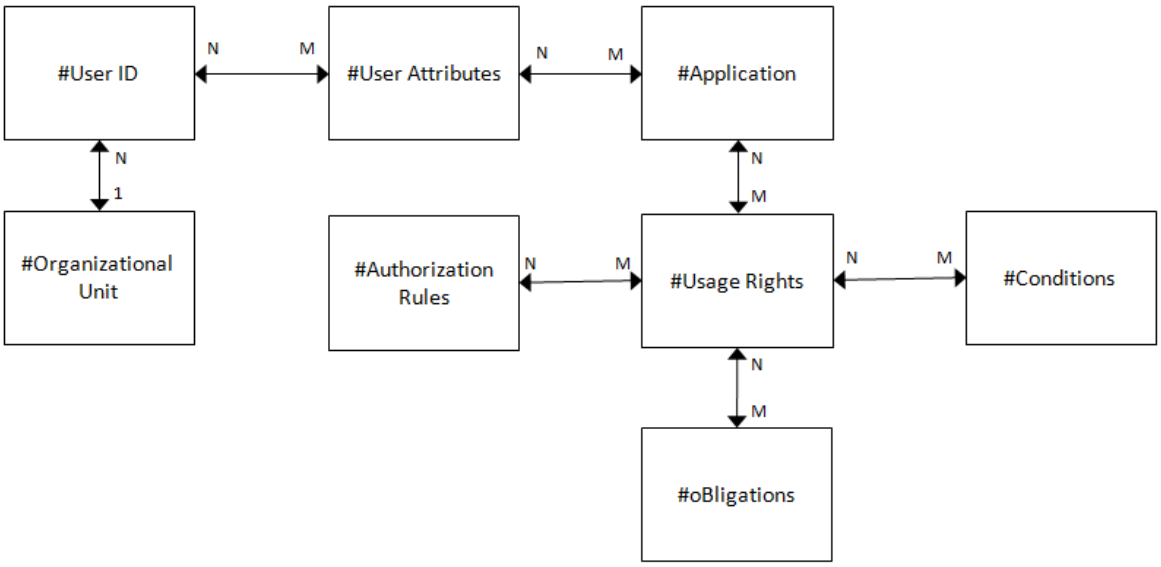

Figure 12. UCON Implementation Data Model.

ACCESS REQUEST

Regarding previous methods, a UCON implementation could Grant Access to previously unknown Users by checking their credentials. With the Request for Usage, the same credentials will be sent to the local UCON Implementation Repository.

In the local UCON repository, the existence of User privileges will be examined; if these do not exist, the Implementation could decide to grant Usage Rights if the Request could be managed locally, or not.

If the request could be managed locally, the UCON implementation will check the Attributes pertaining to the user requesting a Security Profile. These attributes would be associated to a number of Usage rights, taking into account the Authorization Rules, the oBligations, and the Conditions regarding the Subject, the Applications, and the Environmental Conditions.

The Rights, Authorization Rules, oBligations and Conditions must meet all the security standards of the Bank, must be able to decide which are the appropriate usage rights to any type of user based on the attributes before, and during the usage. And after the usage, the attributes of a Subject can change depending on how he (or she) made use of them, in order to allow for better decision-making in the subsequent accesses (See Figure 13).

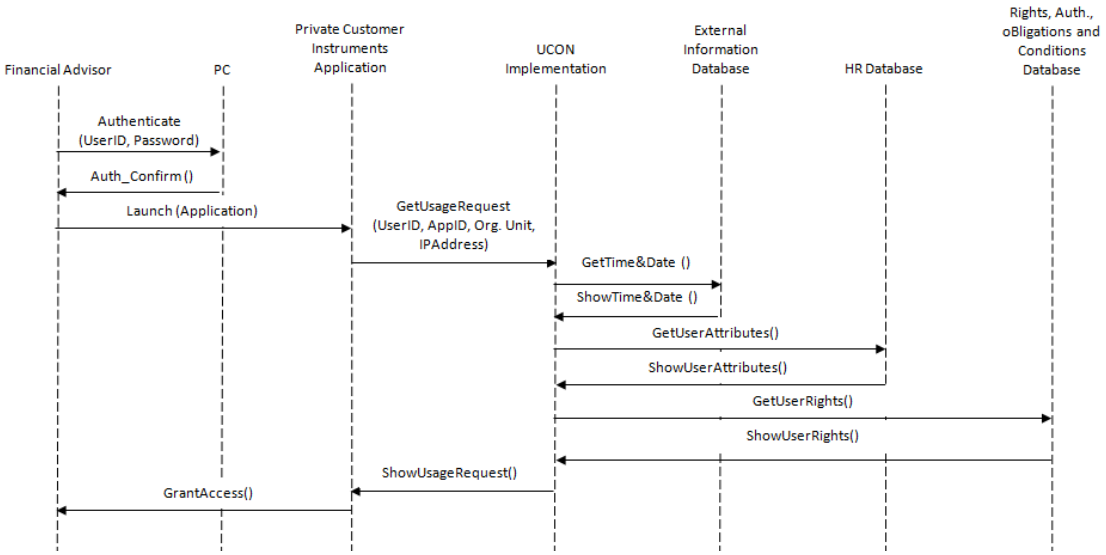

Figure 13. Access Request in UCON Implementation.

If the local UCON implementation could not handle the request, said request would be sent to a central implementation where the same process would take place in a larger scale.

If the *status* attribute of a Subject changed (e.g. the employee called-in sick), the change of that attribute would generate more changes regarding the Usage Control, as the duties of that employee would be distributed among other fellow employees according to fixed company policies. The UCON Implementation would evaluate this Attribute change from the other Subjects, and the modified Usage Rights will be available to the other Subjects upon request. Once the status attribute of the employee came back

to normal, the Usage Rights would be reevaluated, in case that any other attribute changed.

If between usages, the critical Attributes do not change, the UCON implementation would not have the need to reevaluate any Usage Rights.

SYSTEM ADMINISTRATION

As it was the case with the ABAC Implementation, in order to maintain the Separation of Duties, the HR Department of the bank would create, modify, or delete the Subjects and the Subject Attributes used to properly Identify the User within the system, this includes the *status* Attribute but excludes any Usage Attribute linked to the Subject, unless explicitly stated, as these are maintained by a different area. The application administrator would assign the access Attributes for each individual application.

The UCON Implementation manager would be in charge of managing the Rights, Authorization Rules, oBligations, and Conditions that control the relationship between Subject Attributes/Object Attributes.

The UCON implementation should be capable of using the robust set of Rights, Authorization Rules, oBligations, and Conditions to create, modify, or delete any Subject Attributes/Object Attributes relationship, as long as it complies with the previously mentioned set. In this way, the activities of the Policy Administrator would be taken over by the Implementation itself (See Figure 13); however, the UCON implementation administrator should resolve any conflict that may arise in the AC implementation.

The set of Rights, Authorization Rules, oBligations, and Conditions ought to be kept in a separate Database, to ensure continuity even after system downtime (See Figure 14).

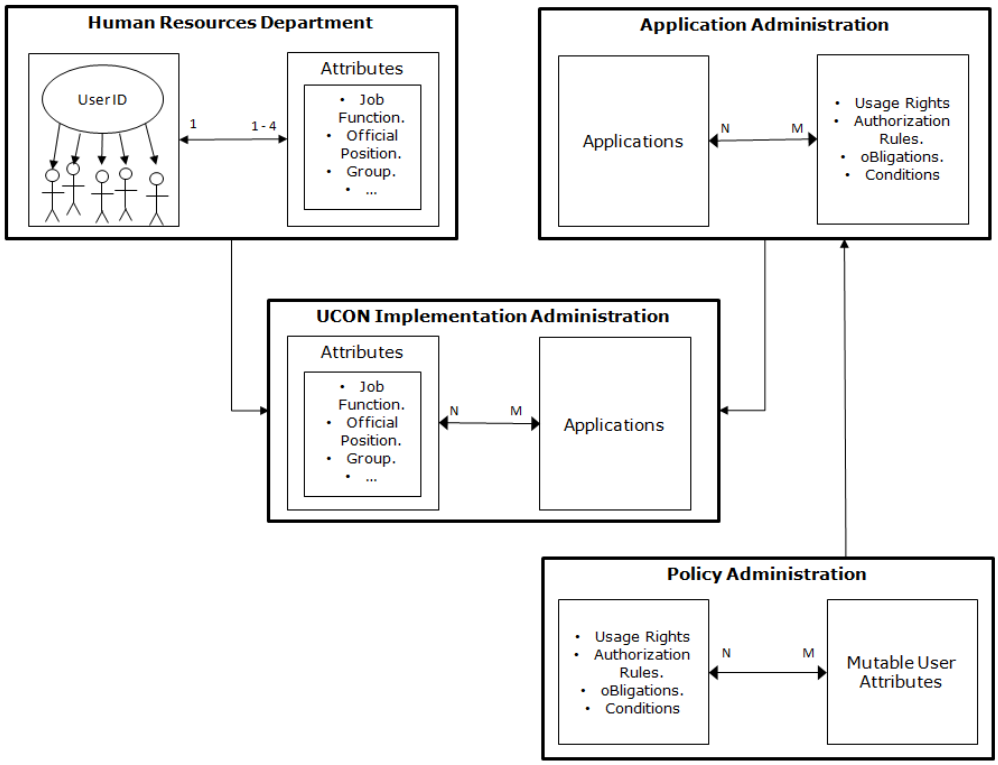

Figure 14. UCON Administration.

DISCUSSION

The pairing solution created in the FUB, regarding Official Position and Job Function could be replaced with external attributes (such as Location and Time) in order to limit the access rights for the users in each branch, thus avoiding any possible security threats.

The performance of each system in the **Entropy Layer** is the following:

- The FUB, and the RBAC system seem too rigid to support temporal and spatial changes. The system behaves as a centralized system, with some applications coming from a mainframe-setup background, instead of client-server or cloud based systems.

- The ABAC implementation allows for a de-centralized architecture, repositories can be distributed across the branches of the bank, and the system does not seem to be as loosely coupled as the FUB/RBAC was, Nonetheless, the system may easily support spatial changes, undergoing through minor changes in the configuration of the AC Implementation.

- The UCON implementation imposes the biggest challenge regarding workload demand, as the AC method is constantly checking and changing attributes, as well as enforcing Access decisions. Coupling more spatial objects in time can be less resource consuming, as the system is readily available for such changes.

In the **Assets Layer**, the behavior is as follows:

- The FUB/RBAC implementation "hosted roles and delivered access rights for usage within other applications that run under various environments such as UNIX derivatives (SINIX/AIX) or WINDOWS NT",

which makes it a diverse environment to set up the AC implementation, meaning that the software barrier had been overcome.

The hardware has to be able to support modern applications, as well as mainframe-based applications. However, any change applied to the system, such as the creation of a role would put the system to the test.

- The ABAC Implementation may behave like that FUB/RBAC implementation regarding the applications used by the bank, but in order to manage the Policies required for the system to work as expected, more applications need to be integrated with the already existing ones.

  Managing the policies may also mean that the hardware requirements could be more demanding, thus increasing the need for systems that can better handle the workload. Unlike the FUB/RBAC implementation, the system is designed to batter handle changes in the elements, entities and distribution of the IS.

  The decision in Figure 7 to exclude the PAP from the rest of the Implementation was made to maintain the Separation of Duties, that already existed within the bank.

- The UCON Implementation ought to behave in the exact same way with the Applications used by the bank in terms of the transference of Access Rights that the Applications need to use (in the implementation the name Access Rights has been changed to Usage Rights). But even in these conditions, the IS of the bank should be prepared to handle more information, and newer software to grant secure and proper Access through this type of AC.

An increase in the amount of information transmission between Subjects and Objects can be expected. A demand for more powerful hardware (in the Headquarters, as well as the branches) should be taken into account, as the system could (at least in theory) resolve most AC conflicts. With conditions changing constantly within the IS, there would be a need to keep the system online 24/7.

Regarding the **Management Layer**, the implementations may perform in the following way:

- From the information found in the paper by Schaad, et al. (2001), the actual layout of the FUB/RBAC Implementation cannot be determined; nonetheless, based on the year of the publication, and the features that make Core RBAC, it can be assumed that the AC model relied on a centralized structure, with Access Terminals on the branches and the FUB in the main Headquarters.

  Having the AC Implementation in one place reduces the management complexity encountered in more distributed systems, as all the efforts regarding Subject, Object, and Policy updates, as well as resolution of conflicts are focused on one location.

  In this particular case, three administrators had to control the Subject Roles, the FUB, and the Applications; with the FUB administrator solving issues with non-standard company affiliation Subjects on top of his already assigned tasks. Schaad, et al. (2001) also specified: "The current access control system does not provide the flexibility to meet these changes without a major administrative effort."

- The ABAC solution previously mentioned could create a number of problems regarding the management of it, as the solution is more

detailed, the difficulties regarding the upkeep of the system can make maintenance more cumbersome.

The decentralization of the system, which is intended to make it more secure and (depending on the tasks performed) faster, could become problematic when handling downtime or policy changes.

In case of an update of the AC policies, the HR Department, the ABAC Implementation Administrator, and the Application administration may need to update the attributes used to determine the profile, potentially increasing the difficulty of making such changes. As policies are updated, these must be tested to check if they do not leave any security inconsistencies.

Being a finer grained solution, permission revoking would be a harder task, requiring extensive debugging, when compared to RBAC.

- The UCON implementation, unlike the ABAC implementation would not need to have a Policy Administration, as the Implementation itself would manage the Usage Rights given a number of predefined Authorization Rules, oBligations and Conditions. Hence, the maintenance of the AC would be centered in Updates.

  Unlike ABAC, policies in the UCON implementation could be changed while the system is online, thanks to the mutability attributes of the AC Method.

  Conflict resolution in UCON would be problematic, as the newly created solution must comply with the existing Rights, Authorizations, oBligations, and Conditions, without leaving further security inconsistencies that could be exploited by unscrupulous hands.

Being a finer grained solution, permission revoking would be a harder task, requiring extensive debugging, when compared to RBAC.

Regarding the **Logic Layer** of the three systems, the following points must be addressed:

- Due to the nature of the business (a bank), an appropriate assumption would be that almost all the applications would be interactive-based, rather than batch-based.

- RBAC and ABAC do not take into account the possible environmental changes in the ecosystem of the bank. This means that there may be a risk for the system to become inefficient.

- On the other hand, UCON is programmed to cope with changes in the environment. As Mutability and Continuity are inherent attributes of the AC model, the risk of inefficiency of the system is lower.

CONCLUSIONS

The variables affecting a firm, such as its size, integration of its IS with suppliers and customers, and the environment that surrounds it determine which type of Access Control Method is more suitable for the firm.

It is expected that a more detailed and/or flexible implementation of an Access Control Method would translate into higher setup costs, as the infrastructure, the security mapping, and the implementations of policies from the firm is more intensive than that found in less flexible Access Control Methods.

UCON could exceed the control that ABAC and RBAC may impose; nonetheless, it is important to state that setting up and maintaining this kind of systems is expensive in terms of time, workforce, and money. The company must evaluate how much security is needed taking into account the instability of the environment, the mission, vision, objectives, and policies of the firm.

Regardless of the Access Control Implementation, careful administration and monitoring of user/role/rights assignment is needed to prevent security violations and conflicts.

The maintenance cost of the AC Implementations cannot be addressed properly, as the information regarding the number of employees needed to maintain the system may vary not only regarding the implementation, but the budget, size, and security policies of the firm may influence the aforementioned costs.

The methodology extracted from Gouglidis & Mavridis (2009) was sufficient to explore the limitations of each implementation.

The central concept when addressing new implementations of the Access Control System was to avoid changing the interaction between employees and IS. This concept shaped the basic structures, data models, AC administrations and access request of the implementations. If the central concept is changed, the outcome could differ from the one shown in this project.

REFERENCES

Coyne, E., & Weil, T. R. (2013). ABAC and RBAC: Scalable, Flexible, and Auditable Access Management. *IT Professional , 15* (3), 14-16.

Crampton, J. (2005, June). A reference monitor for workflow systems with constrained task execution. *In Proceedings of the tenth ACM symposium on Access control models and technologies* , 38-47.

Ferraiolo, D. F., & Kunh, D. R. (1992). Role-Based Access Controls. *15th National Computer Security Conference* , 554-563.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Informationand Systems Security , 4* (3), 224-274.

Fugini, M. (2012, October). Corporate Information Systems - Notes from the class. Como, CO, Italy.

Gouglidis, A., & Mavridis, I. (2009). A Foundation for Defining Security Requirements in Grid Computing. *13th Panhellenic Conference on Informatics* , 180-184.

Hu, V. C., & Scarfone, K. (2012). *Guidelines for Access Control System Evaluation Metrics.* National Institute of Standards and Technology.

Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Miller, R., & Scarfone, K. (2014). *Guide to Attribute Based Access Control (ABAC) Definition and Considerations* (Vol. 800). NIST Special Publication.

Jin, X., Krishnan, R., & Sandhu, R. (2012). A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC. *Data and Applications Security and Privacy XXVI* , 41-55.

Ma, Z., Huang, J., Yang, Y., & Niu, X. (2013). Verifiable Threshold Authorization for Scalable and Secure Digital Rights Management. *Journal of Software , 8* (6), 1526-1535.

Merriam-Webster Inc. (2014, May 7). *Hierarchy - Definition and More from the Free Merriam-Webster Dictionary*. Retrieved May 7, 2014, from Merriam-Webster: http://www.merriam-webster.com/dictionary/hierarchy

Park, J., & Sandhu, R. (2004). The UCON(ABC) Usage Control Model. *Transactions on Information and System Security , 7* (1), 128-174.

Park, J., & Sandhu, R. (2002, June 3-4). Towards Usage Control Models: Beyond Traditional Access Control. *SACMAT* .

Sandhu, R. S., & Coyne, E. J. (1996). Role-Based Access Control Models. 38-47.

Sandhu, R. S., & Samarati, P. (1994). Access Control: Principle and Practice. *IEEE Communications Magazine* (9).

Schaad, A., Moffett, J., & Jacob, J. (2001, May). The Role-Based Access Control System of a European Bank: A Case Study and Discussion. *Proceedings of the Sixth ACM symposium on Access Control models and Technologies* , 3-9.

Wei, Q., Crampton, J., Beznosov, K., & Ripeanu, M. (2008, June). Authorization Recycling in RBAC Systems. *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies* , 63-72.

Yuan, E., & Tong, J. (2005, July). Attributed Based Access Control (ABAC) for Web Services. *Proceedings of the IEEE International Conference on Web Services (ICWS'05)* .

Zhang, X., Parisi-Pressice, F., Sandhu, R., & Park, J. (2005). Formal Model and Policy Specification of Usage Control. *ACM Transactions on Information and System Security (TISSEC) , 8(4)*, 351-387.