Declaring and Enforcing Users' Control in Personal Data Servers. A case study of UCON$_{ABC}$ model.

Athanasia Katsouraki

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Informatics

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, Heraklion, GR-700 13 , Greece

University of Paris Sud XI
Computer Science Department
Batiment 650, P.C. FR-91405, Orsay cedex, France

Thesis Advisor : Prof. *Luc Bouganim*

This work has been performed at the **University of Paris-Sud XI,Computer Science Department, Orsay.**
The work is partially supported by the **Inria Rocquencourt.**

University of Paris-Sud XI
Computer Science Department

Declaring and Enforcing Users' Control in Personal Data Servers.
A case study of UCON$_{ABC}$ model.

Thesis submitted by
Athanasia Katsouraki
in partial fulfillment of the requirements for the
Masters' of Science degree in Informatics

THESIS APPROVAL

Author : ______________________
Athanasia Katsouraki

Committee approvals : ______________________
Luc Bouganim
Professor, Thesis Supervisor, Committee Member

____
Alexandre Allauzen
Assistant Professor,Research director, Committee Member

____
Fatiha Sais
Assistant Professor, Committee Member

Paris, June 2013

Abstract

Innovations in information technology impose new security and privacy issues which include traditional access control, trust management (TRM) and digital right management (DRM). Traditional access control has focused on a closed system where all users are known and primarily utilizes a server-side reference monitor within the system. TRM covers privacy issues such as the authorization for strangers in an open environment, like the Internet. DRM has focused on client-side control of digital information usage. By unifying the aforementioned three areas, UCON offers a promising approach for the next generation of access control. In this thesis, we briefly introduce the state of the art on Access Control, including some Access Control Models. Subsequently, we describe in details the UCON$_{ABC}$ model which is a Usage Control Model. We propose two implementations of this model (UCONengine). More precisely, the first implementation is in Java, using the Schema of eXtensible Access Control Markup Language (XACML) and the second implementation is in Java, using Data structures and a Database. We demonstrate several examples to show the power of this model. Finally, we present some conclusions and future work.

Keywords :
Usage Control, Access Control, Digital Right Managent, Trust Management, UCON.

Résumé

Les innovations liées aux nouvelles technologies de l'information constituent une révolution scientifique et technique de grande ampleur avec de nouveaux enjeux liés à la protection de la vie privée et à la sécurité. Ceci englobe le contrôle d'accès traditionnel, le contrôle d'usage, la gestion de confiance (TRM) et la gestion des droits numériques (DRM). Dans le contrôle d'accès traditionnel, les utilisateurs sont connus à l'avance et un moniteur de référence côté serveur est utilisé. Au contraire, avec la gestion de confiance, les autorisations sont gérées dans un environnement ouvert dans lequel les utilisateurs ne sont pas connus, comme c'est le cas de l'Internet. La gestion des droits numériques se concentre sur le contrôles, côté client, pour l'utilisation de contenus numérique. En unifiant les trois domaines précités, UCON offre une approche nouvelle et prometteuse. Dans ce rapport de stage de master, nous présentons tout d'abord les modèles classiques de contrôle d'accès, le modèle UCON$_{ABC}$ puis proposons deux mises en œuvre de ce modèle (UCONengine) en Java. Nous concluons et présentons quelques pistes pour des travaux futurs.

Mots clefs :
Contrôle d'accès, Contrôle d'usage, Gestion de droits digitaux, Gestion de confiance, UCON

Περίληψη

Οι καινοτομίες που έχουν λαβει χώρα στην τεχνολογία των πληροφοριών επιβάλουν τη συστηματική μελέτη νέων ζητημάτων ασφαλείας και ιδιωτικότητας. Ωστόσο, απαιτείται να συνδυαστούν στοιχεία από τρεις περιοχές, οι οποίες είναι οι ακόλουθες:
Traditional Access Control, Trust Management, Digital Right Management.
Η περιοχή του Traditional Access Control επικεντρώνεται σε ένα κλειστό σύστημα, όπου όλοι οι χρήστες είναι γνωστοί και αρχικά υλοποιεί ένα server-side reference monitor στο πλαίσιο του συστήματος. Η περιοχή του Trust Management εισάγεται για να καλυφθεί η άδεια για τους ξένους χρήστες σε ένα ανοιχτό περιβάλλον όπως το διαδίκτυο. Η περιοχή του Digital Right Management ασχολείται με τον client-side έλεγχο της χρήσης της ψηφιακής πληροφορίας. Με την ενοποίηση των τριών αυτών περιοχών, το UCON αποτελεί μια πολλά υποσχόμενη προσέγγιση για την επόμενη γενιά του ελέγχου πρόσβασης. Καλύπτει τόσο ένα κεντρικά ελεγχόμενο περιβάλλον, καθώς επίσης και ένα περιβάλλον όπου η κεντρική αρχή ελέγχου δεν είναι διαθέσιμη. Σε αυτή την εργασία, αρχικά παρουσιάζουμε κάποια θεωρητικά θεμέλια για τον έλεγχο πρόσβασης (Access Control), τον έλεγχο χρήσης (Usage Control), καθώς επίσης περιγράφουμε κάποιες γενικές έννοιες του μοντέλου UCON. Στη συνέχεια, ορίζουμε τα στοιχεία του μοντέλου και επίσης περιγράφουμε τις δυο υλοποιήσεις της μηχανής UCON, με τις οποίες και ασχοληθήκαμε. Αξίζει να αναφερθεί πως καθόλη την εργασία παρουσιάζουμε πολλά παραδείγματα για να δείξουμε την δύναμη του μοντέλου αυτού. Τέλος, εκθέτουμε ορισμένα συμπεράσματα και ανοιχτά θέματα για μελλοντική δουλειά.

Λέξεις Κλειδιά:
Έλεγχος Χρήσης, Έλεγχος Πρόσβασης, Ψηφιακή Διαχείριση Δικαιωμάτων, Αξιοπιστία Διαχείρισης, UCON.

Acknowledgements

First of all, I would like to thank the members of the SMIS project, who made me feel as a part of the team since the beginning and guided me through the postgraduate program. The last ten months of my life I have been more productive than ever in my whole life, and my team's behavior played an instrumental role in that.

Especially for this thesis, I would like to thank Mr. Philippe Pucheral who offered me this position at INRIA Roquencourt. Furthermore, I would like to thank my supervisor Mr. Luc Bouganim and Mr. Nicolas Anciaux, whose choices on occurring challenges were decisive. I would like to mention that our collaboration has been extremely pleasant. Without their support and guidance my Master's Thesis would not be the same.

I would like to acknowledge the financial support of the INRIA Roquencourt. I would like to thank the staff of the INRIA for helping me in several ways all these months.

I would like to thank also the University of Crete which gave me the opportunity to be a part of it through the Greek-French Postgraduate Program. More specifically, I would like to thank Mr. Vasilis Chrisotphides for the great things that he has taught me through his courses, for his support and guidance for Greek-French Postgraduate Program and for his recommendation letter to my SMIS-team, as well. Furthermore, I would like to thank Mr. Xenophon Zabulis and Mr. Konstantinos Panagiotakis for their recommendation letters.

Last, but not least, I would like to thank my family for their encouragement and precious support during my studies. I would also like to thank my friends and people that are close to me for their company and support. I would like to dedicate this work to my family and to the whole SMIS-team.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Access control in computer and information security is concerned with authentications and authorizations. It determines tasks and activities that legitimate and authenticated subjects can perform in order to access resources in a system. Therefore, protection of the resources of a system against undesired user access could be crucial issue that is examined by Access Control.

There is a variety of models that has been developed and they have evolved greatly. Research, related to Database Security has shown that the existing traditional access control models such as DAC, MAC and RBAC are not suitable to cover the needs of modern computing environments. The limitations of traditional access control are evident in distributed computing environments. More specifically, the traditional access control models and systems were designed for closed systems where the identities of subjects or users were well known before hand.

UCON [1, 2] is a conceptual framework for controlling access to objects. UCON encompasses traditional access control, trust management, or DRM and privacy issues and goes beyond in its definition and scope. UCON achieves fine-grained control on digital resources regardless of their locations and serves as an extension of the traditional access control models, including new features in order to cover the challenges of modern, open dynamic distributed computing environments. In UCON the authorizations to access objects are based on predicates that utilize subject and object attributes to define the requirements and conditions under which a subject is allowed to access an object. The attributes of the subject or an object may change as a result of which the state of the system changes. The distinguishing aspects of UCON that sets it apart from the prevailing access control models is the continuity of access decision and attribute mutability. It was designed to provide effective protection in modern digital computing environments and distributed systems. UCON demonstrates a major shift in the design of access control models. UCON provides either to the provider or the owner of digital resources the access control of their resources even after they have been disseminated to domains outside of their direct control.

UCON$_{ABC}$ model is a model that can be used to realize access control scenarios of traditional access control, trust management and DRM as well. In traditional access control, the decision for accessing the objects depends on the authorizations of the subject. However, in UCON, the authorizations are determined by subject and object attributes and a policy defined by the owner of the object.

The UCON model consists of eight core components which can be separated into the following categories: *Main components of UCON*, *Decision Factors of UCON* and *Decision Properties of*
UCON. The main components of UCON model include subjects which are entities which are associated with subject attributes, and they hold and exercise certain rights on objects, objects which are entities which are associated with object attributes, either by themselves or with rights and rights which are privileges that a subject could hold on an object. Subsequently, decision factors of UCON model include authorizations which are functional predicates that should be evaluated for usage decision, obligations which are functional predicates that verify mandatory requirements that a subject should perform before or during a usage and conditions which are a set of decision factors that the system should verify during the authorization process along with authorizations before allowing usage of rights on a digital object. Finally, decision properties of UCON model include mutability of attributes and and continuity of decision access. The former provides UCON$_{ABC}$ the flexibility to accommodate complex access control scenarios where the attributes of subjects and objects may change as a result of access to certain resources. The latter enables the evaluation of authorizations and conditions either before access and/or during the usage of the object, by the subject. It allows the system to revoke access from the object as soon as the requirements for access to the object are not met.

For instance, Athanasia (subject) works at Computer Science Department (subject attribute) and she is responsible for designing the UCON$_{ABC}$ model (subject attribute). This Department has a folder that contains images (objects) which can be viewed by her under some specific conditions that have to be fulfilled. More precisely, Athanasia (subject) has to belong to the Computer Science Department (condition) in order to have the permission for accessing (right: 'read') "image1.jpg" (object), 3 times (mutability) from this folder and each time she can access it for 5 seconds (condition). Thus accessTime of image may be changed for Athanasia as a result of access to "image1.jpg".

This thesis was accomplished in the two cycles of my internship on Database Security Domain at the SMIS team at INRIA-Rocquencourt; cycle one (September 2012 to February 2013) and cycle two (February 2013 to June 2013). During cycle one, we have studied and demonstrated the state of the art on Access Control, including some Access Control Models and their limitations in the real-world systems. Subsequently, having reviewed a significant part of the related literature, we have illustrated the conceptual UCON$_{ABC}$, a model for Usage Control of individuals data and we have proposed an implementation of UCON$_{ABC}$ model (UCON engine) as a general engine that could be used into an application (i.e. social network application) or as a service within a system. This engine has been implemented in Java, using PolicySet, Rules and Schema of eXtensible Access Control Markup Language (XACML), due to several advantages. A noticeable advantage could be that XACML policy language is used to express access control rules and conditions, providing flexibility. Finally, during cycle two, we have developed the UCON$_{ABC}$ model using Data structures and a Database in order to extend and simplify the aforementioned implementation.

Outline This thesis is organized as follows. Chapter 2 briefly introduces the state of the art on Access Control, including some Access Control Models. UCON$_{ABC}$, a Usage Control Model is described in Chapter 3. Chapter 4 demonstrates a Java implementation of UCON$_{ABC}$ model (UCON engine), using the Schema of eXtensible Access Control Markup Language (XACML). Chapter 5 demonstrates a Java implementation of UCON$_{ABC}$ model (UCON engine), using Data structures and a Database. Finally, we conclude in Chapter 6 and discuss future work.

Chapter 2

Access Control

Access control is a process of mediating each request to resources and data which maintained by a system and determining whether the request should be granted or denied [3]. An Access Control model defines a policy and provides a description of the security properties of the system. In this model, a user or a process (subjects), are entities that would like to access objects. Objects are the entities that contains the information that a user or a process (subject) would like to access.

An access control policy defines the rules where they can regulate access to resources, determining which user is allowed to access which objects. A system is considered secure when the access control mechanism implements an access control model and when we are able to prove its security.

2.1 Traditional Access Control Models

Access control models are often classified as either discretionary or non-discretion ary. The three best-known models include the Discretionary Access Control model (DAC), Mandatory Access Control model (MAC), and Role Based Access Control model (R-BAC), as well.

2.1.1 Discretionary Access Control

Discretionary access control [4, 5] needs the user's identity and define some authorization rules in order to determine which are the user's privileges within a system. The owner of resources determines the policy which is related to the decision of which users are allowed to access which object, what kind of access and what are the privileges that they have [6]. The discretionary model is represented by Access Control Matrix (ACM) [7], which is either a structure or a table where a subject can be represented on each row and an object can be represented on each columns. The entries of the table's cells represent the rights that a subject can exercise on an object.

Two important concepts can be identified in DAC [4, 5]:

1. **File and data ownership**
   Every object in the system has an owner. In most cases in a DAC system, the initial owner of an object constitutes the subject that caused it to be created. As far as the access policy for an object is concerned, it is determined by its owner.
2. **Access rights and permissions**
   Access rights and permissions constitute the controls that an owner of the objects can assign to other subjects, for particular resources. As far as access controls are concerned, they may be discretionary in Access Control List (ACL).

One of the major weaknesses of DAC is that programs inherit the identity of the invoking user. This weakness makes the system vulnerable to malicious programs. There is no real security on the flow of information in a system. For instance, let us assume a user Athanasia who gives access to Catherine to one of her images. Then, Catherine can view the image and also can edit one of her images, including the image of Athanasia. Furthermore, Catherine is able to grant access to a third user Charlie even he is someone that Athanasia had denied access to the original image before. However, from now on Charlie can access the image of Athanasia.

Another weakness of DAC lies on that only the owner can determine which object could be accessed or not by which subject. There are no constraints that are related to the usage of information that the user has received. Finally, the owner could not be able to revoke the access, since there are no specific constraints (i.e. temporal, spatial) to be used for limiting the usage of the object.

Unix, Linux, Windows access control is based on DAC. More precisely, the Unix file mode that represent write, read, and execute for each of User, Group and Others. As another example, capability systems provides discretionary controls since they permit subjects to transfer their access to other subjects.

2.1.2 Mandatory Access Control

In Mandatory access control [8, 9, 10] there is a labeling mechanism which is employed in order to label an object in a system. Thus, MAC is able to control the flow of information between different objects within a system. For instance, the subjects and objects that belong to a system should have labels that are assigned to them. The sensitivity label of the subject and object specifies their level of trust. A security policy is specified in order to determine the flow of information between objects which have specific labels and can be enforced according to some specific security requirements of an organization. An owner of an image does not determine the permissions for an object.

Two methods [8, 10] are used for applying mandatory access control:

1. **Rule-based access control**
   This type of control defines some specific conditions for accessing the re-

quested object(s). A Mandatory Access Control system implements a simple form of rule-based access control in order to determine if the access should be either granted or denied by comparing the object's and subject's sensitivity label.

2. **Lattice-based access control**
   A lattice model is a mathematical structure that defines greatest lower-bound and least upper-bound values for a pair of elements, such as a subject and an object.This type of control can be used for a complex access control decisions involving multiple objects and/or subjects.

Some examples of MAC models include the Bell La-padula, Biba, Clark Wilson, Chinese wall policy, etc. Few systems implement MAC. SELinux which is a NSA research project, added a MAC architecture to the Linux Kernel. TOMOYO Linux is a lightweight MAC implementation for Linux and Embedded Linux, developed by NTT Data Corporation. SUSE Linux and Ubuntu 7.10 added a MAC implementation called AppArmor. Oracle Label Security is an implementation of MAC in the Oracle DBMS.

2.1.3 Role-Based Access Control

Role-based access control (R-BAC) [11, 12, 13] is an access policy which simplifies the specification of authorization rules by grouping the users of a system into roles, according to their duties and authorizations within the organization.

A role in RBAC can be viewed as a set of permissions.
There are three primary rules [13] that are defined for RBAC:

1. **Role assignment**
   If the subject has selected or been assigned a role, then it can execute a transaction.
2. **Role authorization**
   If the subject has selected or been assigned a role and the subject's active role is authorized for the subject, then the user can only take on the roles for which s/he is authorized.
3. **Transaction authorization**
   A subject can execute a transaction if the transaction is authorized for the subject's active role. In the previous rule, if the subject has also selected or been assigned a role and the subject's active role is authorized then this rule ensures that the user can execute only transactions for which s/he is authorized.

It can improve significantly the administration and management of the permissions which are assigned to the users. Some permissions are given to roles. The members who belongs to a role inherit the permissions that are assigned to the role and the decision for access is based on the role of a user. The permissions which are associated with a role can also be changed according to the changes in the organizational structure.

R-BAC is a non-discretionary model, such as MAC, but it handles the permissions differently. MAC controls I/O permissions based on a user's clearance level and additional labels. In R-BAC collections of permissions that include complex operations such as an e-commerce transaction are controlled. R-BAC differs also from DAC, since DAC allows users to control access to their resources, in constrast to R-BAC, where the access is controlled at the system level, outside of the user's control.

R-BAC examples include commercial applications and also military systems, where there are also multi-level security requirements. Systems including Microsoft Active Directory, Microsoft SQL Server, SELinux, grsecurity, FreeBSD, Solaris, Oracle DBMS, PostgreSQL 8.1, implement some form of R-BAC.

2.2 Modern Access Control and Digital Rights Management

Research in the Access Control domain has demonstrated that access control needs enhancements in order to meet the needs of modern applications and systems. New concepts which include Trust Management and Digital Rights Management (DRM), has been arisen. In this Section, we are going to discuss the above concepts.

2.2.1 Trust Management

Authentications and authorizations of users, computers, electronic devices and networks that rely on user authentication and access control, are widely used. This is insufficient for open environments, such as the Internet, since as an open environment, it requires flexible and dynamic access control. In addition, distributed systems lack central control and also its users can not be predetermined by any means. Trust management was developed in order to fill the gaps and enhance the traditional access control.

In a distributed computing environment, the amount of people who request a resource is quite large. In the majority of web services it is common that the subjects are unknown to the system prior to the access request. In Trust Management [14], there is no need of a predefined identity in order to authorize the subject. The authorizations are based on the credentials of the subject rather than a predefined identity, such as a username or/and a password. The credentials of a subject are checked in order to decide if access is either granted or denied to the subject, according to the security policy of the system.

Trust Management systems include those systems that has focused on authentication [15], general purpose authorizations [14, 16], and those that were developed to cover specific needs [17, 18].

2.2.2 Digital Right Management

Digital rights management (DRM) [19, 20, 21, 22, 23] addresses the information security that is needed to some businesses which are dealing with digital content (i.e books, music, images, and videos).

DRM allows to restrict the access to copyright digital content [19, 20] which is being distributed to the clients. The usage of the object is controlled by a client side reference monitor. This monitor is responsible for protecting against copyright violation and also allows only the legal users to use the copyrighted material according to some specific conditions which are defined by its owner.

DRM systems' examples [19] include those which have been developed by various organizations for commercial purposes and are concerned with the protection of intellectual property rights. For instance, it worths to mention the Fair Play by Apple, the Windows media DRM by Microsoft and the Adobe's protected streaming. However, there is no compatibility between the aforementioned examples. For instance, Microsoft oriented devices are not able to play the digital material which is protected by Apple's FairPlay. This issue complicates and affects the integration of DRM solution into some mainstream digital devices that are used by the users who are the consumers of digital material.

Chapter 3

Usage Control

In Chapter 2, we demonstrated some well known Access Control Models, such as the Discretionary Access Control Model, the Mandatory Access Control Model and the Role-Based Access Control Model. Although they have some important features for the access control, these models have the difficulty in addressing the needs of the modern information systems. One of the main reasons is that the traditional Access Control models have focused on authorization only. The authorization decision is made only before access is allowed. There is no ongoing control concept to be taken into consideration, no further enforcement is possible during the access and rights are not supported. A subject can hold granted rights for non defined period of time. However, this is not acceptable for obligation-based or condition-based controls as well as for the other dynamic authorization controls.

Most of the modern information systems often require more than authorizations, such as consumable attributes or rights (i.e credit balance in a transaction or a limited number of usages). For instance, a user may have to fill out a certain form or s/he may have to click a button ("yes" button) in order to get a license agreement for usage allowance. These required actions are called *Obligations* and have to be fulfilled by subjects for usage allowance. Furthermore, most of the digital objects that exist can be played or distributed only on a certain device or location. These environmental restrictions are called *Conditions* and have to be fulfilled for access.

3.1 The UCON Model and the Reverse UCON model

The notion of usage control is able to integrate obligation or/and conditions. The usage control has been modeled in the UCON$_{ABC}$ model. The overview of the UCON$_{ABC}$ model that we explain here is inspired from [1, 2]. In this Section, we are going to describe the UCON model and the reverse form of this model.

UCON$_{ABC}$ model consists of eight core components that are involved in the authorization process (see Figure 3.1). We can separate those core components into parts. *Main components of UCON* comprise subjects, subject attributes, objects, object attributes and rights. *Decision Factors* comprise authorizations, obligations and conditions. These are functional predicates that have to be evaluated for the usage decision. *Decision Properties* include mutability of attributes and continuity of decision access.

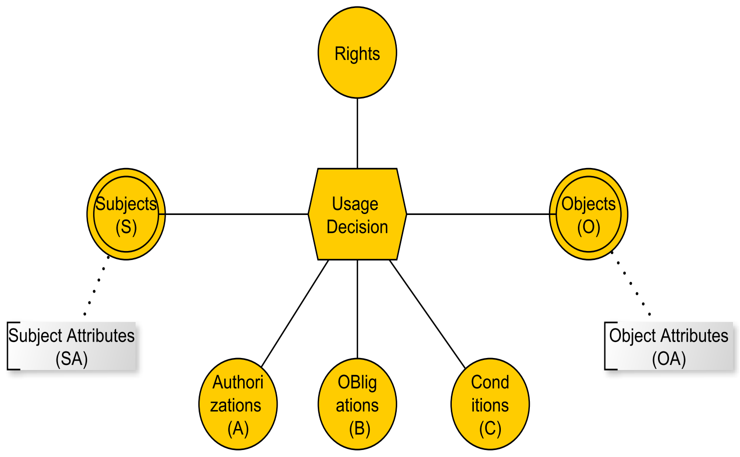


Figure 3.1: UCON$_{ABC}$ Model Components

By obtaining or exercising usage rights on a digital object, such as on an image, mp3 file or a video file, then another digital information object may be created. This new object is called derivative object, includes privacy-related information and needs controls for its access and usage, as the original object. Thus, the usage control is reversed, since the provider subject becomes now the consumer subject.

Let us consider Athanasia, who is the consumer subject (CS), who would like to watch an avi movie (see Figure 3.2). In order to obtain watch rights, she should agree on payment-per-watch, which is an obligation (OB). She should provide information of her credit card. When she exercises the watch rights, she has to report her usage log on the video, which is also an obligation (OB). After payment, Athanasia is both a provider subject (PS) and identifiee subject (IS) of the log/payment information and may hold certain rights (PR and IR) on them (i.e she can delete her ID from log). Both payment information and log information constitutes derivative objects. The distributor which could be a video production company, may have rights to collect log information either by putting an obligation on consumer rights or by giving consumer rights to get some store credits on log reports. If Athanasia has rights to get some store credit based on her watch time, then it is distributor's obligation to issue certain credit to Athanasia.

3.1.1 Main Components of UCON model

As we discussed in Section 3.1, the main components of UCON include subjects, subject attributes, objects, object attributes and rights. In this section, we are

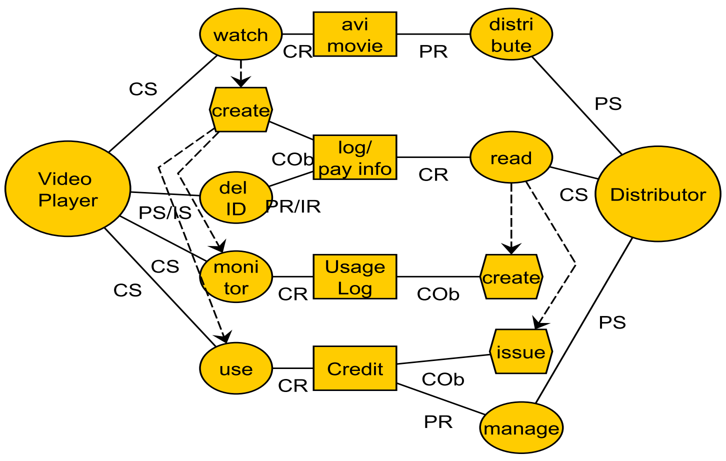


Figure 3.2: The reverse UCON model.

going to describe deeply these components.

Subjects are entities which are associated with attributes, and they hold and exercise certain rights on objects. For instance, a subject inludes a user, a group, a role, or a process. In UCON$_{ABC}$, the subjects could be either consumer subjects (CS), which receives rights and objects and use those rights in order to access the objects, provider subjects (PS), which provides an object and hold certain rights on it, or identifiee subjects (IS), who are identified in digital objects, including privacy-sensitive information. For instance, CS, PS, IS include an e-book reader, an author of an e-book, a patient of a health care system, respectively.

Subject attributes are properties of the subjects that can be used for the authorization process. For instance, subject attributes include identities, roles, credits, etc.

Objects are entities which are also associated with attributes. In UCON$_{ABC}$, objects could be either privacy sensitive or privacy non-sensitive and original or derivative. In the former case, privacy sensitive object includes individually identifiable information that cause privacy problems if it is not used in an appropriate way. In the latter case, as we discussed in Section 3.1 a derivative object means derived or cited from an original work in order to create another digital work.

Similarly to subject attributes, the ***object attributes*** include certain properties that can be used for the authorization process. Subjects can hold rights on objects, whereby the subjects can access or use objects. For instance, object attributes include security levels, ownerships, etc.

Rights are privileges that a subject could hold on an object. They consist of a set of usage functions which are able to enable the access of a specific subject for specific objects. For instance, a usage could be watching a movie. The authorizations of rights require associations with subjects and objects. Similarly to subjects and objects, rights can also be divided into consumer rights (CR), provider rights

(PR), and identifiee rights (IR).

3.1.2 Decision Factors of UCON model

As we discussed in Section 3.1 decision factors include authorizations, obligations and conditions. In this section, we are going to describe deeply these components.

Authorizations are functional predicates that should be evaluated for a usage decision. They return if the subject is allowed or not to perform the requested rights on the object. Authorizations evaluate subject attributes, object attributes and requested rights together with a set of authorization rules for the usage decision. Authorizations could be either pre-authorizations (preA) which are performed before a requested right is exercised or ongoing-authorizations (onA) which are performed while the right is exercised. Generally, most of traditional access control policies, such as MAC, DAC, RBAC and Trust Management utilize some form of pre-authorization for their decisions.

Obligations are also functional predicates. They verify mandatory requirements that a subject should perform before or during a usage. Obligations could be either pre-obligations (preB) which can utilize some kind of history functions in order to check whether some specific activities have been fulfilled or not ("true/false") or ongoing-obligations (onB) which should be satisfied continuously or periodically while the allowed rights are in use.

Conditions are a set of decision factors that the system should verify during the authorization process along with authorizations before allowing usage of rights on a digital object. Condition predicates evaluate current environmental status in order to check if relevant requirements are met or not ("true/false"). These predicates can not be mutable, since conditions are not under direct control of individual subjects. Evaluation of conditions can not update any subject or object attributes. For instance, some examples of condition requirements include accessible time, location, etc. Subject and object attributes are used in order to choose the condition requirements that should be used for a request. However no attribute is included within the requirements themselves.

3.1.3 Decision Properties of UCON model

As we discussed in Section 3.1 decision properties include mutability of attributes and continuity of decision access. In this section, we are going to describe deeply these components.

Continuity of Access Decision constitutes a feature of UCON$_{ABC}$ model which is responsible for enabling the evaluation of authorizations and conditions not only before access but also during the usage of the object when a subject is exercising the rights and permissions are granted to subject on an object. For instance, the provider of a video file restricts this file to be accessed for 70 seconds only, for advertising purposes. If someone would like to watch the whole movie, the user should pay a subscription fee. If the user has not paid for the subscription and accesses the file, then the system keeps track of the time since the file has been open and should revoke access as soon as the time duration of 70 seconds expires.

Mutability of attributes constitutes a feature that provides UCON$_{ABC}$ the flexibility to accommodate complex access control scenarios that are met in modern computing environments. This refers to the process of change in the values of subject, object or environment attributes as a result of access to an object. For instance, Athanasia would like to read an e-book and should pay for that. When she accesses the digital content of e-book, then her balance reduces. The permissions of subjects and the state of the system are affected by changes in the values of subject, object and environment attributes.

3.2 UCON$_{ABC}$ Core Models

R. Sandhu and J. Park [1, 2] have developed a family of core models for usage control, based on three decision factors: authorizations, obligations, and conditions, along with continuity of access decision and mutability of attributes in order to support modern access control requirements, enhancing the domain of access control. These models have focused on the enforcement process ("core" models), excluding any administrative issues.

If all attributes are immutable, no updates are possible as a consequence of the decision process. Section 3.1.3 demonstrated that mutability of attributes allows certain updates either on subject or object attributes as side effects of usages. Thus, for mutable usage, updates are required either before (pre), during (ongoing), or after (post) the usage. Based on these criteria, they have developed 16 possible model spaces for usage control (all the possible combinations).

Figure 3.4, Figure 3.5, Figure 3.6 demonstrate each of UCON$_A$, UCON$_B$ or UCON$_C$, the three basic models. Figure 3.3 illustrates some possible combinations of UCON$_{ABC}$ models. It is a graphical representation of the richness of the model space available in the UCON$_{ABC}$ family.

3.3 Basic UCON$_{ABC}$ Models

In this section, we are going to describe each of the three UCON$_{ABC}$ models.

3.3.1 UCON$_A$ Models

UCON pre-Authorizations Models

In UCONpre$_A$ models [1, 2], an authorization decision process is done before usage is allowed. preA examines usage requests using subject attributes, object attributes, and rights and then decides whether the request is allowed or not.Three detailed models exist based on mutability variations. pre-updates

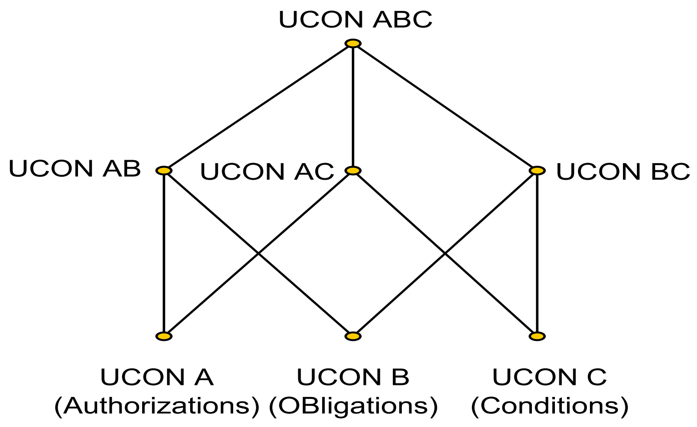


Figure 3.3: The UCON$_{ABC}$ Family of Core models.

and post-updates on subject and object attributes are optional procedures to perform update operations on them (see Figure 3.4).

UCON ongoing-Authorizations Models

In UCONon$_A$ models [1, 2], usage requests are allowed without any 'pre' decision-making and authorization decisions are made continuously or repeatedly while usage rights are exercised. The currently allowed usage right is revoked when a certain requirements become dissatisfied, as a result its exercise is stopped. Four detailed models exist (see Figure 3.4).

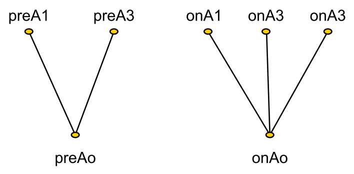


Figure 3.4: The UCON$_A$ models.

3.3.2 UCON$_B$ Models

UCON pre-Obligation Models

In UCON$_{preB}$ models [1, 2], pre-obligations should be fulfilled before access is permitted. preB is a kind of history function that checks whether certain obligations have been fulfilled or not and return true or false for the usage decision (see Figure 3.5).

UCON ongoing-OBligations Models

In UCON$_{onB}$ models [1, 2], usage requests are allowed without any 'pre' decision-making. By ongoing authorizations, monitoring is actively involved in usage decisions while a requested right is exercised (see Figure 3.5).

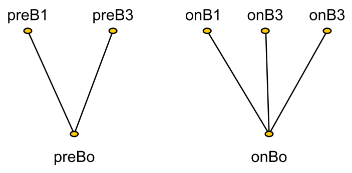


Figure 3.5: The UCON$_B$ models.

3.3.3 UCON$_C$ Models

Generally, the UCON$_C$ models cannot be mutable. This is different from the fact that the value of conditional status can be changed as the environmental situation is being changed (i.e. current time is changed as time goes, or a wireless access point is changed as a user moves around a building). However, subject or object attributes are not used for usage decision process but they are used in order to decide what kind of condition elements (preCON) have to be enforced for usage decision.

UCON pre-Conditions Models

In UCON$_{preC}$, conditions constitute environmental restrictions that should be satisfied for usages. Generally, preCON are environmental restrictions that are not related to subjects and objects (see Figure 3.6).

UCON ongoing-Condtions Models

Environmental restrictions have to be satisfied while rights are in active use. This could be supported within UCON$_{onC}$ model [1, 2]. In this model, usages are allowed without any decision process at the time of requests and there is an ongoing conditions predicate in order to check certain environmental status repeatedly throughout the usages, as well (see Figure 3.6).

Figure 3.6: The UCON$_C$ models.

3.4 Examples

In this section we demonstrated how the traditional access controls, as well as the modern access controls such as MAC, DAC, R-BAC and DRM that we presented in Chapter 2 can be realized within UCON$_{preA0}$ and UCON$_{preA1}$.

Example 1. Mandatory access control (MAC) can be realized within UCONpre$_{A0}$
:
L is a lattice of security labels with dominance $\geq$
clearance : S $\rightarrow$ L, classification : O$\rightarrow$ L
ATT(S) = clearance, ATT(O) = classification
allowed(s, o, read) $\Rightarrow$ clearance(s) $\geq$ classification(o)
allowed(s, o,write) $\Rightarrow$ clearance(s) $\leq$ classification(o)

Example 2. Discretionary access control (DAC) using Access Control Lists [1] can be realized within UCONpre$_{A0}$:
N is a set of identity names
id : S $\rightarrow$N, one to one mapping
ACL : O $\rightarrow$ 2^{NxR} ,n is authorized to do r to o
ATT(S) = id
ATT(O) = ACL
allowed(s, o, r) $\Rightarrow$ (id(s), r) $\in$ ACL(o)

Example 3. A Digital Rights Management (DRM) example of preUpdate is payment-based access. (Pay-per-use with a pre-paid credit (UCONpre$_{A1}$))
M: is a set of money amount
credit: S$\rightarrow$M
value: O x R $\rightarrow$ M
ATT(s): credit
ATT(o,r): value
allowed(s,o,r) $\Rightarrow$credit(s) $\geq$ value(o,r)
If subject holds enough credits then it uses certain rights on specific objects.
preUpdate(credit(s)): credit(s) = credit(s) - value(o,r)
A subject's credits reduced by the value if usages at the time of request approval

3.5 Related work

Many studies have been done on the field of access control. In this thesis, we proposed the design and the implementation of the conceptual UCON$_{ABC}$ model [1, 2] as a general purpose engine. Nowadays, there are various technological devices, such as smartcards, personal digital assistants (PDAs), mobile phones and personal computers, that are able to share and distribute digital information and computational resources. This variety of heterogeneous devices are usually network connected and use some services, such as Clouds. Technological innovations have raised several new and important challenges in protecting digital resources.

1. ACL is a functional mapping of object to multiple ids and rights

In [24], an authorization framework based on standards like XACML and SAML for distributed systems is proposed. This paper has been the starting point and motivated us for studying XACML standards on detail in order to express some parts of the model in the initial implementation (see Chapter 4).

In [25], Damiani et al. identify the requirements for access control in open environments, such as the Internet. They have been proposed some extensions for the access control models (i.e attribute based access control model, semantics aware access control model). In [26], Sastry et al. proposed an example for dynamic creation and interpretation of attributes in multiple systems. A Context Attribute-Based Access Control model (CABAC) is proposed in [27]. M. Covington and M. Sastry proposed the specification of the authorization policies, including the attributes which do not involve either the subjects or objects. Therefore, they proposed the Transaction Attributes. Those attributes constitute the subject's attributes which come up from a transaction.

Although the UCON model constitutes a very powerful model, it has some limitations. For instance, as we mentioned access control to a resource is being controlled either before an access is permitted (preC), or continuously (onC). This is called contuinity of decisions. However, since a subject (s) requests to use an object (o), then the usage control decision is based on either information related to subject and/or object. In addition, no information which is related to previous or/and current usages, as well as the previous requested usages that were denied is kept. For instance, an attribute update can be executed only if the requested usage is allowed. Subsequently, the feature of UCON which is related to the attribute mutability does not support a policy rule that is based on historical information, since no history is recorded.

In [28] Grompanopoulos et al. having noticed the UCON's limitations, proposed a Usebased Usage CONtrol (UseCON) model that constitutes an extension of UCON model, providing an enhanced utilization of the usage decision criteria and also an enchanced support for complex usage modes, taking into account current usages exercised in the system.

Chapter 4

Case Study and Model Implementation 1

This work has focused on the implementation of the UCON model, as we described it in Chapter 3. The implementation has been done in java using PolicySet, Rules and Schema of eXtensible Access Control Markup Language (XACML), due to several advantages. A noticeable advantage could be that XACML policy language is used to express access control rules and conditions, providing flexibility. We designed and implemented the UCON model as a general engine that could be used into an application (i.e. social network application) or as a service within a system (i.e. reference monitor).

4.1 Design

This section demonstrates the UCON engine which was developed as an engine that could be used as a part of an application or as a service within a system, since it decides about the usage of an object.

In order to implement the UCON engine, we have distinguished the different parts of the UCON model (see Figure 4.1). The *Authentication Manager* is able to authenticate the subject, based on subject's ID and subject's password. The *Access Manager* is designed in order to manage and regulate access to objects by processing subjects' requests which are related to exercising rights on objects. It decides if the access is going to be granted or denied, as well.

The policy evaluator allows this decision to be taken, while the *Context Manager* is focused on keeping track of the active usage sessions on an object. The feature of UCON$_{ABC}$ model which is known as *Continuity of Access Decision* is illustrated by Context Manager. The *Primitive Actions Manager* is designed for the updates of the attributes. The *Attribute Update Manager* is developed as the representation of the *Mutability of Attributes* of the UCON model.

The attributes are stored and updated in XML files which contains subject and object attributes. The *Attribute Reader* allows retrieving subject and object

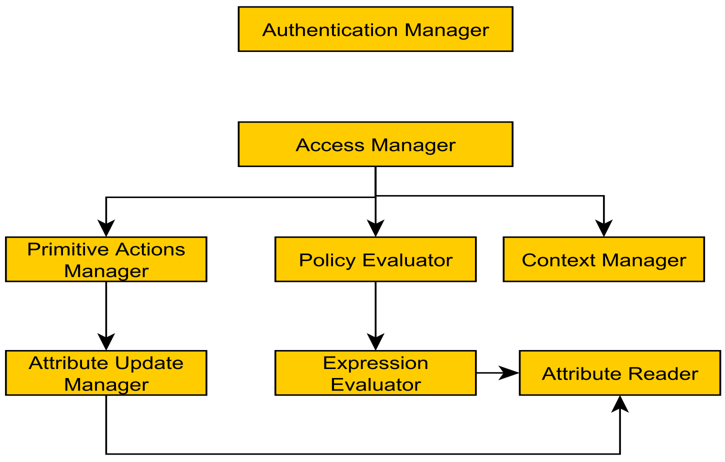


Figure 4.1: Modules of UCON engine.

attributes. The values of these attributes are very important for the evaluation of expressions during the evaluation of policy and the attribute update process, as well. The *Policy Evaluator* is developed for retrieving and parsing of the policies which are associated with the usage of an object. The *Expression Evaluator* evaluates the expressions that are fundamental for the evaluation of usage policies within a policy.

There is a log file (see Figure 4.2) where the information for the usage of the object and enforcement of the UCON policy (XML structure) is kept. It contains all those entries which are related to the usage sessions along with the attribute updates which are taken place within a usage session.

The information which is kept for an event includes the *Start of Usage*, the *End of Usage*, the *State Transition* and the *Attribute Update*. The information for the *Start of Usage* is needed in order to identify the start of a new usage session. This information is recorded in the log file along with the time stamp, the subject's id, the object's id and the right which subject has requested to exercise on the object. Subsequently, the information about *End of Usage* is needed in order to identify the end of a usage session. This information is recorded in a log file along with the time stamp and consists of the subject's id, the object's id and the right which a subject has requested for a specific object. It is needed to keep information which is related to the state transition in the log file, in case of a state transition includes either the time stamp or the previous state. We should keep in the log file *time stamp*, *type of the attribute update* (preUpdate, postUpdate, or onUpdate), *the object's ID*, *the new value of the object attribute* and *the previous value of the object attribute*, in order to verify the attribute updates.

```
<?xml version="1.0" encoding="UTF-8"?>
<log:UsageLog xmlns:log="ucon/UsageLog"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="ucon/UsageLog/UsageLog.xsd ">
       <log:UsageSessions>
             <log:NewUsageSession Object="Document" Right="Read"
                        Subject="athanasia001" TimeStamp="2013/04/04 16:48:09"/>
                        <log:AttributeUpdate AttributeName="NumberOfReadings"
                         OldValue="[0.0]" TargetObjectID="Document"
                         TimeStamp="2013/04/04 16:48:09" UpdateType="PreUpdate"
                         UpdatedValue="[1.0]">
                            <log:ExpressionEvaluated FunctionID=
                              "internship.katsurak.model.algorithms.functions.Add"
                                   Result="[1.0]" TimeStamp="2013/04/04 16:48:09">
                                   <log:Inputs>
                                   <log:Attribute AttributeName="NumberOfReadings"
                                    ObjectID="Document" Value="[0.0]"/>
                                             <log:Constant>1</log:Constant>
                                   </log:Inputs>
                            </log:ExpressionEvaluated>
                         </log:AttributeUpdate
                   <log:StateTransition CurrentState="Requesting"
                    PolicyEvaluated="Requesting" PreviousState="NotApplicable"
                    SessionID="athanasia001,Document,Read"
                    TimeStamp="2013/04/04 16:48:09"/>
                   <log:StateTransition CurrentState="Accessing"
                     PolicyEvaluated="PermitAccess"
                     PreviousState="Requesting"
                     SessionID="athanasia001,Document,Read"
                     TimeStamp="2013/04/04 16:48:10"/>
       <log:EndUsageSession Object="Document" Right="Read" Subject="athanasia001"
            TimeStamp="2013/04/04 16:49:28"/>
       </log:UsageSessions>
</log:UsageLog>
```

Figure 4.2: Part of usage log file.

4.2 UCON Policies

The policies in the UCON model control the access on an object by defining conditions. Each transition of a state is allowed under some specific conditions. A UCON policy imposes the attribute updates which will be carried out at each state during a usage session. The policies are specified by the object's owner (subject) or within a document. They include conditions and attribute updates that are predicates to indicate the condition under which a state transition is allowed. The primitive actions consist of the attribute updates that will be performed at each state.

The policies in the UCON model are referred to a set of policies for each state. For instance, a state could be *requestingAccess, grantedAccess, denied Access, revok edAccess and endAccess*, according to the model's description in Chapter 3. Figure 4.3 illustrates the different policies that can be executed. In every policy, there is a target that identifies the subject, the object and the right for which the policy is going to be enforced.

The requesting policy includes the PreUpdates that will be performed since the requesting state is made. The transition from requesting state to accessing state is controled by a policy. This policy includes the conditions under which a subject can access an object for a specific right. When the conditions in this policy are met in the requesting state, the subject's state is the accessing state and attribute updates associated with this state are performed. The aforementioned updates are the OnUpdates and are continuously performed during the subject is accessing the object.

When the conditions are not met in requesting state in order to access the object then the access denied state occurs. The attribute updates associated with this state are performed. As long as a subject is allowed access to an object then the conditions for accessing the object are checked continuously. When the conditions can no longer be met due to some changes, then the access is revoked. The attribute updates that are associated with the revoked state are performed. The subject could stop accessing the object before access is revoked and the endAccess state happens and the associated attribute updates are performed.

As we discussed before, there are four states in which a subject could be transited, including *requesting state*, *accessing state*, *denied state* and *revoked state*. There are various policies that can be defined for the usage of an object. Those policies are written in XML, according to the Schema of eXtensible Access Control Markup Language (XACML).

More precisely, the *RequestingAccess Policy* is the type of policy that consists of the attribute updates that are to be performed when the requesting state happens. The *PermitAccess Policy* is the type of policy that consists of the conditions that should be met before accessing state is allowed to happen. The evaluation of this policy is taking place during the accessing state. When the conditions are no longer met, then the access is revoked. The *Accessing Policy* is the type of policy that includes the updates of attributes that are performed during the accessing state. The *AccessDenied Policy* is the type of policy that includes the updates of the attributes that are performed in the access denied state. The *RevokeAccess Policy* is the type of policy that includes the updates of attributes that are performed in the access revoked state. Finally, the *EndAccess Policy* is the type of policy that contains the attribute updates that are performed after the user ends the access.

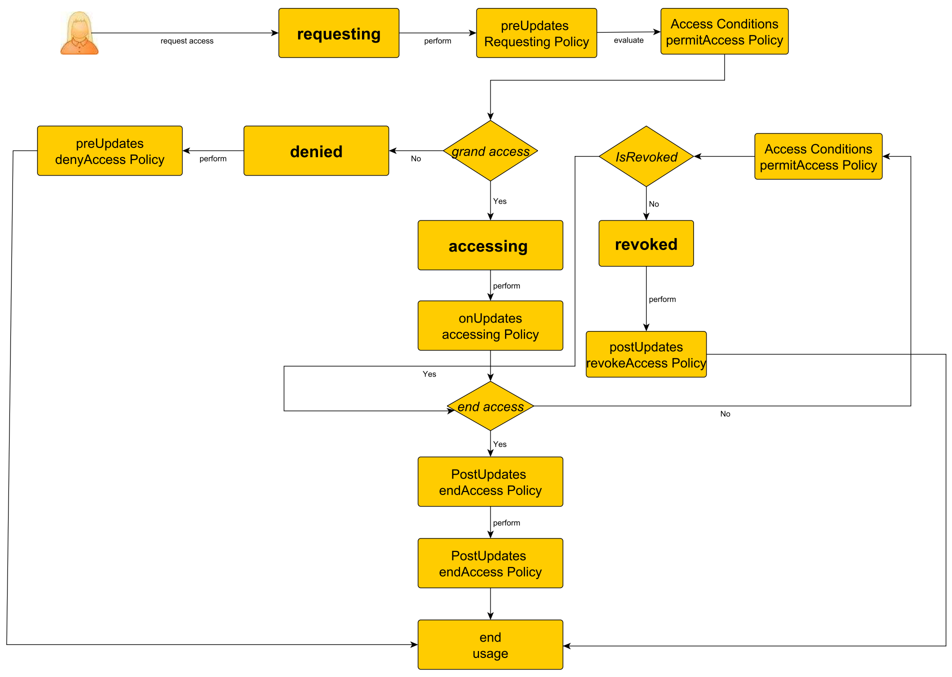

request access
requesting
perform
preUpdates Requesting Policy
evaluate
Access Conditions permitAccess Policy
preUpdates denyAccess Policy
perform
denied
No
grand access
Yes
accessing
perform
onUpdates accessing Policy
IsRevoked
Access Conditions permitAccess Policy
No
revoked
perform
postUpdates revokeAccess Policy
Yes
end access
No
Yes
PostUpdates endAccess Policy
perform
PostUpdates endAccess Policy
end usage

Figure 4.3: Control flow and execution of different policies.

A schema for the UCON policy has been developed. We included inside the policy file the *Target of Policy*, the *Type of Policy* (see Figure 4.4), the *Conditions*, the *Primitive Actions*, the *Attribute Updates* and some Expressions. The target of the policy, as we mentioned above, identifies the subjects, objects and the rights for which the policy is applicable. The policy type determines the state of the usage session at which the policy is evaluated.

The condition element is used to determine the conditions under which the state transition into accessing state is allowed. This is determined by expressions. The primitive actions include attribute updates for implementing the concept of mutability of attributes, as we discussed in Chapter 3. Finally, some expressions are used in the definition of conditions and attribute updates in a policy.

4.3 Scenarios

In this section, we are going to illustrate by an example the power of UCON engine and demonstrate how UCON models can be applied to protect the privacy of digital information.

This example can be explained with Figure 4.5, Figure 4.6 and Figure 4.7 that represent the subject, object and policies under which the access is regulated, accordingly. Suppose a user (Athanasia) (see Figure 4.5) would like to exercise the right "Read" on an object (Document) (see Figure 4.6). If the policies in the

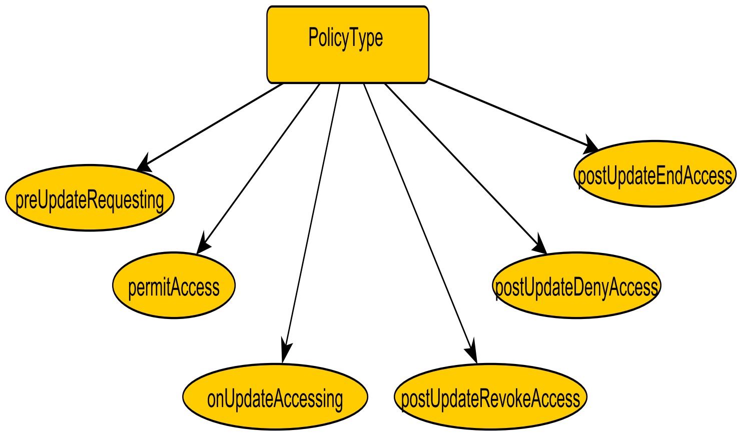


Figure 4.4: Type of Policy.

policy file (see Figure 4.7) are satisfied successfully, then Athanasia can access the Document.

The user Athanasia has a unique subjectID and subjectPWD in order to authenticate herself. An AuthenticationManager is responsible for her authentication. A Context Manager maintains a Map of subjectID (key) which allows storing the subjects' and objects' attribute that will be accessible to the subject and the corresponding rights (in this case: "Read"). The decisions are received from the AccessManager (i.e. through events). Reading and Updating of attributes are performed by the AttributeManager.

In this case, Figure 4.5 demonstrates the structure of a subject (Athanasia) with a unique subjectID (athanasia001) and a unique subjectPWD (SecretCode). We defined the subject's attributes as follow: "Department" with value "Computer-Science, "DesignerOf" with value "UCON model", "Name" with value "Athanasia" and "DocumentReadTime" with value "12:00".

Similarly, we present the structure of the object (Document) with a unique objectID (Document) (see Figure 4.5). We defined the object's attributes as follows: "TimeForRead" with value "12.00" and "NumberOfTimes" with value "2.0".

Figure 4.7 represents the policies that have to be fulfilled. According to the policy file, Athanasia can read the Document, if she belongs to Computer Science Department, from twelve o'clock until one o'clock. The ExpressionEvaluator evaluates the expressions in a policy. The PolicyEvaluator evaluates the policies and attributes updates. When the Context Manager requests for an object then the Access Manager creates a decisionPolicyEvent and the mutabilityEvent. The conditions are checked continuously. If the conditions for accessing the object either have changed, or can no longer be met, then access is revoked.

```
<?xml version="1.0" encoding="UTF-8"?>
<obj:Object xmlns:obj="ucon/Object"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="ucon/Object/Object.xsd ">
       <obj:ObjectID>athanasia001</obj:ObjectID>
       <obj:Attributes>
              <obj:Attribute>
                     <obj:AttributeName>Department</obj:AttributeName>
                     <obj:AttributeValues>
                     <obj:AttributeValue>ComputerScience</obj:AttributeValue>
                     </obj:AttributeValues>
              </obj:Attribute>
              <obj:Attribute>
                     <obj:AttributeName>DesignerOf</obj:AttributeName>
                     <obj:AttributeValues>
                     <obj:AttributeValue>UCON model</obj:AttributeValue>
                     </obj:AttributeValues>
              </obj:Attribute>
              <obj:Attribute>
                     <obj:AttributeName>Name</obj:AttributeName>
                     <obj:AttributeValues>Athanasia</obj:AttributeValues>
              </obj:Attribute>
              <obj:Attribute>
                     <obj:AttributeName>DocumentReadTime</obj:AttributeName>
                     <obj:AttributeValues>
                            <obj:AttributeValue>
                                   1200
                            </obj:AttributeValue>
                     </obj:AttributeValues>
              </obj:Attribute>
       </obj:Attributes>
       <obj:Secret>MySecretCombinationElias</obj:Secret>
</obj:Object>
```

Figure 4.5: XML file with subject's information.

```
<?xml version="1.0" encoding="UTF-8"?>
<obj:Object xmlns:obj="ucon/Object" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="ucon/Object/Object.xsd ">
        <obj:ObjectID>Document</obj:ObjectID>
        <obj:Attributes>
                <obj:Attribute>
                        <obj:AttributeName>TimeForRead</obj:AttributeName>
                        <obj:AttributeValues>
                                <obj:AttributeValue>1200</obj:AttributeValue>
                        </obj:AttributeValues>
                </obj:Attribute>
                <obj:Attribute>
                        <obj:AttributeName>NumberOfTimes</obj:AttributeName>
                        <obj:AttributeValues>
                                <obj:AttributeValue>2.0</obj:AttributeValue>
                        </obj:AttributeValues>
                </obj:Attribute>
        </obj:Attributes>
</obj:Object>
```

Figure 4.6: XML file with object's information.

```
<?xml version="1.0" ?>
<p:PolicySet xmlns:p="ucon/UCONPolicy"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="ucon/UCONPolicy/UCONPolicy.xsd ">
       <p:Policy PolicyType="PermitAccess">
              <p:Target>
                     <p:Subject>athanasia001</p:Subject>
                     <p:Object>Document</p:Object>
                     <p:Right>Read</p:Right>
              </p:Target>
              <p:Condition CombiningAlgorithm=
              "internship.katsurak.model.algorithms.combiningalgorithms.All">
                     <p:Expression FunctionID=
                     "internship.katsurak.model.algorithms.functions.LogicalAnd">
                            <p:Expression
                            FunctionID=
                            "internship.katsurak.model.algorithms.functions.StrEqua
                                   <p:ObjectAttribute>
                                   <p:RequestingSubject ></p:RequestingSubject>
                                   <p:AttributeName>Department</p:AttributeName>
                                   </p:ObjectAttribute>
                                   <p:Constant DataType="String">ComputerScience
                                   </p:Constant>
                            </p:Expression>
                            <p:Expression
                            FunctionID=
                            "internship.katsurak.model.algorithms.functions.LogAnd"
                          <p:Expression
                              FunctionID= "internship.katsurak.model.algorithms.
                                      functions.TimeGreaterThan">
                         <p:Expression
                             FunctionID="internship.katsurak.model.algorithms.
                                        functions.EnvironmentVariable">
                            <p:Constant DataType="String">CurrentTime </p:Constant>
                        </p:Expression>
                                      <p:Constant DataType="Time">1200</p:Constant>
                       </p:Expression>
                       <p:Expression
                            FunctionID="internship.katsurak.model.algorithms.
                                       functions.TimeLessThan">
                        <p:Expression
                             FunctionID="internship.katsurak.model.algorithms.
                                            functions.EnvironmentVariable" >
                            <p:Constant DataType="String">CurrentTime </p:Constant>
                       </p:Expression>
                               <p:Constant DataType="Time">1300</p:Constant>
                                   </p:Expression>
                                     </p:Expression>
                                      </p:Expression>
              </p:Condition>
       </p:Policy>
</p:PolicySet>
```

Chapter 5

Case Study and Model Implementation 2

In Chapter 4, we have focused on the designation and the implementation of the UCON$_{ABC}$ model as a general engine and we proposed an implementation which is based on the description of the model [1, 2] by Jaehong Park, using the Schema of eXtensible Access Control Markup Language (XACML) [29, 30, 31] and XML [32, 33, 34, 35, 36, 37, 38].

In this Chapter, we demonstrate a new perspective on the implementation of the UCON$_{ABC}$ model, using Data structures and a Database in order to extend and simplify the aforementioned implementation. We have proposed an application for Data Sharing (i.e. load/edit images) in which we used the new UCON Engine that we developed using SQL [39, 40, 41, 42, 43] and java. This application could be a part of a Social Network, in which the user can view or edit the images of other users (right: load or edit), according to the permissions of the owner of the image. In our case, a SQL Schema, some functions which represents the UCON Engine have been developed and a graphical user interface (GUI) has been designed.

5.1 Design

In this section, we propose a SQL schema, as an approach to simplify the previous implementation of Chapter 4.

The UCON engine consists of the SQL Schema and some functions. Those functions check the conditions that have to be fulfilled and decide if the access is going to be granted or denied. A location Database keeps pairs of IPs and country prefixes, in order to include contextual information to our conditions and a graphical user interface (GUI) has been designed. Our goal has been to develop an application, in which the UCON Engine is embedded in order to show the power of this model, through some scenarios.

The Data Sharing Application is a general engine for individuals data protection. People have been embarrassed by putting too much information on the internet. This application could be a part of Social Network, in order to protect individuals data. This application was developed in Netbeans IDE 7.2.1. In terms of Database, we used Apache Derby DB due to the following advantages. First of all, Apache Derby is easy to install, deploy, and use. Furthermore, it is based on the Java, JDBC, and SQL standards and provides us an embedded JDBC driver that helps us to embed Derby in our work.

Our proposed SQL Schema consists of tables which are related to the subjects and the objects which are associated with attributes, the rights that subjects can exercise on objects, the policy rules that should be satisfied, mutation features and continuity of decisions features.

In the rest of this section, we present the SQL schema, the UCON database, the Location database and the functions that regulate the access, in detail.

5.1.1 SQL Schema and UCON Database

We have used Apache Derby DB in order to create the tables of the UCON$_{ABC}$ model.

SubObjCont: We defined the SubObjCont table, for subjects (s), objects (o), context (c) (see Table 5.1). A subject requests to exercise some rights on an object. A subject could be a human or a device acting on behalf of a human. An object is the source that a subject would like to access (i.e. a printer or a file). A context could be an environmental variable (i.e Location of a Subject).

In the Table 5.1, socID indicates the id of subject, object or context, socType indicates whether the column contains subject, object or context, socName is the name of a subject, object or context and a socDescription is additional information for each of them. In our UCON Database, we keep information for the subject (socType), s1 (socID), Athanasia (socName), who belongs to Computer Science Department (socDescription). Similarly, the object (socType), o1 (socID), "image1.jpg" (socName) from folder "meta/objects/" is also kept in the UCON Database. We also keep contextual information (socType), c1 (socID), for Location (socName), with the Description: "France" (socDescription).

socID	socType	socName	socDescription
s1	subject	Athanasia	ComputerScienceDepartment
o1	object	image1.jpg	meta/objects
o2	object	image2.jpg	meta/objects
c1	context	Location	France

Table 5.1: SubObjCont Table.

SubObjAttributes: In Chapter 3, we have mentioned that subjects and objects are associated with attributes, information related to subject and object (i.e.

the Department of a user or the access time for the object). We defined the SubObjContAttributes table, for the subjects' and objects' attributes (see Table 5.2). A subject or an object could have more than one attributes, either mutable or immutable.

In the Table 5.2, socID indicates the id of subject or object that is associated with this attribute, attName and attValue indicate the name and the value of this attribute. In our UCON Database, information for the attributes of subject s1 (socID) includes Department, Specialization, login (attNames) with values ComputerScience, UCONmodel, athanasiakat (attValues), accordingly. Similarly, accessMilliSeconds (time in milliseconds that the object can be accessed) and accessTime (how many times the object can be accessed) are the attributes of objects o1 and o2 (socIDs), with values 5000, 0 and 3000, 0, , respectively. Both attributes are mutable, as their values change as a consequence of the access. For instance, the value for the accessTime increases when the object is accessed.

socID	attName	attValue
s1	Department	ComputerScience
s1	Specialization	UCONmodel
s1	login	athanasiakat
o1	accessMilliSeconds	5000
o1	accessTime	0
o2	accessMilliSeconds	3000
o2	accessTime	0

Table 5.2: SubObjAttributes Table.

Rights: We defined the Rights table, for the type of rights (see Table 5.3).

In the Table 5.3, rID indicates the id of a right, rName is the name of the right and rDescription provides additional information for a specific right. In our UCON Database, the possible rights could be 'w' or 'r' (rIDs) that represent the write (edit the image) or read (load the image) (rNames), accordingly.

rID	rName	rDescription
r	read	LoadAnImage
w	write	WriteAnImage

Table 5.3: Rights Table.

Policy Rule: We defined the PolicyRule table where the policy rules are determined.

In the Table 5.4, pID indicates the id of a policy rule, pName is the name of the policy rule, pDescription is additional information related to the policy rule and rID, sID, oID indicate the ids of right, subject, object, respectively. The policy rule imposes the rule under which a subject could access the object with a specific right. In our UCON Database, information for the policy rules p1 (pID) includes the AccessLoad (pName), subject s1 (sID), right 'r' (rID) that the subject could exercise to the object o1 (oID). Similarly, p2 (pID) includes the AccessEdit (pName), subject s1 (sID), right 'r' (rID) that the subject could exercise to the object o2 (oID). Therefore, the subject s1, could read but not write the object o1 and could write but not read the object o2.

pID	pName	pDescription	rID	sID	oID
p1	AccessLoad	LoadAnImageFromAthanasia	r	s1	o1
p2	AccessEdit	EditAnImageFromAthanasia	w	s1	o2

Table 5.4: PolicyRule Table.

Predicate: We defined the Predicate table which has comparisons between a name of an attribute and a specific value (see Table 5.5). This table is essential for checking some conditions.

In the Table 5.5, prID indicates the id of predicate, socID is the id of either subject or object, attName which is the attribute of the subject or object that is going to be compared with a specific value (prValue), using a comparison operator (compOp). In our UCON Database, information for pr1 (prID) includes the subject's id (socID) s1, its attribute Department (attName), '=' (compOp) and the predicate value ComputerScience. Similarly, pr2 and pr3 include information related to the accessTime for each of images, o1 and o2.

prID	socID	attName	compOp	prValue
pr1	s1	Department	=	ComputerScience
pr2	o1	accessTime	<=	7
pr3	o2	accessTime	<=	2

Table 5.5: Predicate Table.

ABC: We defined ABC table in order to declare the type of ABC model components, such as authorizations (i.e preAuthorizations, ongoingAuthorization), oBligations (i.e preoBligations,ongoingoBligations), conditions (preConditions, onConditions), along with the policy rule and the predicate (see Table 5.6).

In the Table 5.6, pID indicates the id of policy rule, abcType represents the type of the ABC components of the model, prID is the id of the predicate and freq is the frequency factor which gives the amount of millisecond that the abcType has to be checked. In our UCON Database, information for preConditions (abcType) includes the id of policy rule (pID) p1, the id of predicate pr1, pr2 and the frequency factor, which in this case is zero. As far as ongoingConditions (abcType) are concerned, the frequency factor (1000) means that the checks of conditions will be done every 1000 milliseconds.

pID	abcType	prID	freq
p1	preCon	pr1	0
p1	preCon	pr2	0
p2	preCon	pr3	0
p2	preCon	pr1	0
p1	onCon	pr2	1000
p2	onCon	pr3	1000

Table 5.6: ABC Table.

Update: We defined the update table for the attributes updates (i.e PreUpdates, onUpdates) that are going to be held (see Table 5.7).

In the Table 5.7, uID indicates the id of update, socID is the id of subject or object, attName is the name of the subject's or object's attribute that will be updated. The value of this attribute will be changed by the uvalue, according to the Operand. In our UCON Database, we keep information for the update u1, u2 (uIDs), where the value of accessTime increases (Operand: '+') by 1 (uValue), when o1 and o2 are accessed.

uID	socID	attName	Operand	uValue
u1	o1	accessTime	+	1
u3	o2	accessTime	+	1

Table 5.7: Update Table.

Mutation: We defined the Mutation table where we keep the update type along with the policy id, the update id and the frequency factor, in order to represent the continuity of decision (see Table 5.8).

In the Table 5.7, pID is the id of policy rule, uID indicates the id of update, mType is the type of update (preUpdate, ongoingUpdate), freq is the amount of millisecond that the mType has to be checked. In our UCON Database, we keep information for the update u1, u2 (uIDs), where the value of accessTime increases (Operand: '+') by 1 (uValue), when o1 and o2 are accessed.

5.1.2 Location Database

In order to extend the functionality of our application adding features related to user's location, as we discussed it in the Section 5.1.1 (i.e context in SubObjCont table), we have created a Location Database, which is also an Apache Derby DB.

pID	uID	mType	freq
p1	u1	preUp	0
p2	u3	preUp	0
p1	u1	onUp	1000
p2	u3	onUp	1000

Table 5.8: Mutation Table.

In this Database we kept a list of pairs of IP addresses and country prefixes (see Table 5.9). We have developed some functions for retrieving the IP address of the subjects and checks if this IP exists in the list of IPs in the Location Database. If so, it returns the prefix of the country, in order to be evaluated by another function, returning the country where the subject is located.

More precisely, two functions have been developed. The former function *returnIP (String cacheHostname)*, that retrieves the IP of the subject and the *returnCountry (String ip, String cacheHostName, Statement stmt, String countrypref)*, which calls the aforementioned function in order to retrieve the country where a subject is located.

Therefore, by retrieving the user's IP, we are able to know about his/her location and also check the Location factor along with the other conditions. For instance, as we mentioned in Section 5.1.1, we have the context c1 where the Location is France. We retrieve socDescription in order to get the name of the Location that we would like to check with user's country. If the user's location matches to socDescription then the preConditions related to context are satisfied, and then access is granted between the subject and the object.

ipAddress	CountryPrefix
215255255	fr
287255255	gr

Table 5.9: Location Table.

5.1.3 Functions of the UCON Engine

In this section, we illustrated all the functions that we have developed in order to implement the UCON$_{ABC}$ model as an Engine that regulates the access on objects. Figure 5.1 depicts the Data Sharing Application and the UCON engine which is embedded into the application. This Application consists of the graphical interface of the Application (application part) and the UCON Engine (ucon engine part), which is a set of functions that regulates the access related to a specific object for a specific subject.

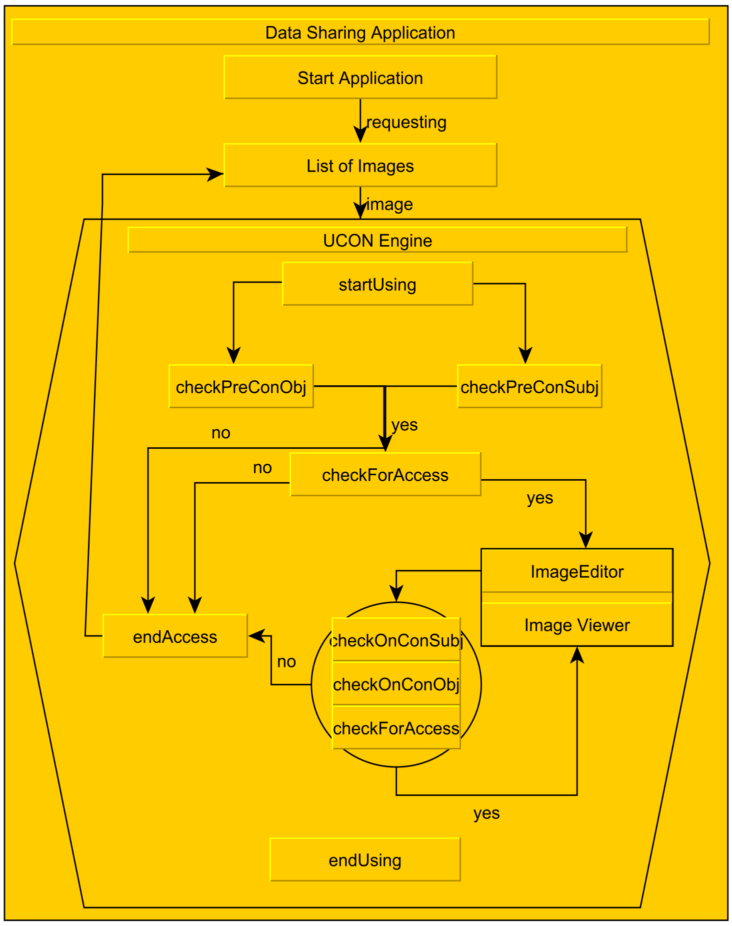


Figure 5.1: Data Sharing Application.

More precisely, the Data Sharing Application has been developed in four parts. The first part consists of the graphical user interface (untrusted part). The second part consists of the manager for the UCON Database which contains all the appropriate functions in order to initialize the Database, create the essential tables, as they were described in Section 5.1.1 and insert values into the tables (Database Part). The third part consists of the functions which check the conditions (preConditions, ongoingConditions) and if the conditions are met, then the UCON Engine gives the permissions to a subject for accessing a specific object (trusted part). It worths mentioning that the conditions are checked not only before the access is granted but also during the access. If the conditions are no longer satisfied, then the UCON engine terminates the access. The fourth part consists of the manager for the Location Database which contains all the appropriate functions in order to initialize the Location Database, create the appropriate tables, as they were described in Section 5.1.2.

5.2 Execution of Data Sharing Application

Firstly, we should initialize the UCON Database. We have developed functions that attempt to connect to the Derby Database and create the tables, as we described in Section 5.1.1. Secondly, we should initialize the Location Database by executing the appropriate functions that connect to the Database and create the location table.

Having initialized our Databases, we should execute the application part. In order to login to the application, the user has to provide his/her username in the appropriate field and then to click the "Login" button (see Figure 5.2). Upon a user is logged in the Data Sharing Application (application part), s/he will be directed to a list of images (application part) which is restricted and s/he could view some or all the images under some specific conditions (see Figure 5.4). In order to view an image, s/he has to click on the image and then press the Load or Edit button.

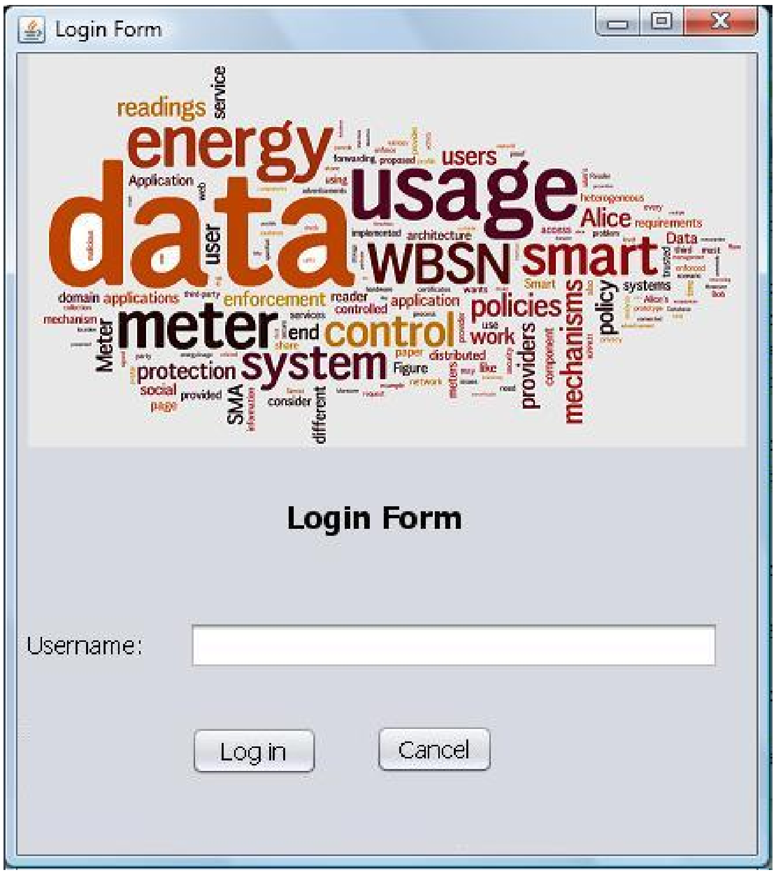


Figure 5.2: Login Page.

Since the user pressed the Load or Edit button, the UCON engine (ucon engine

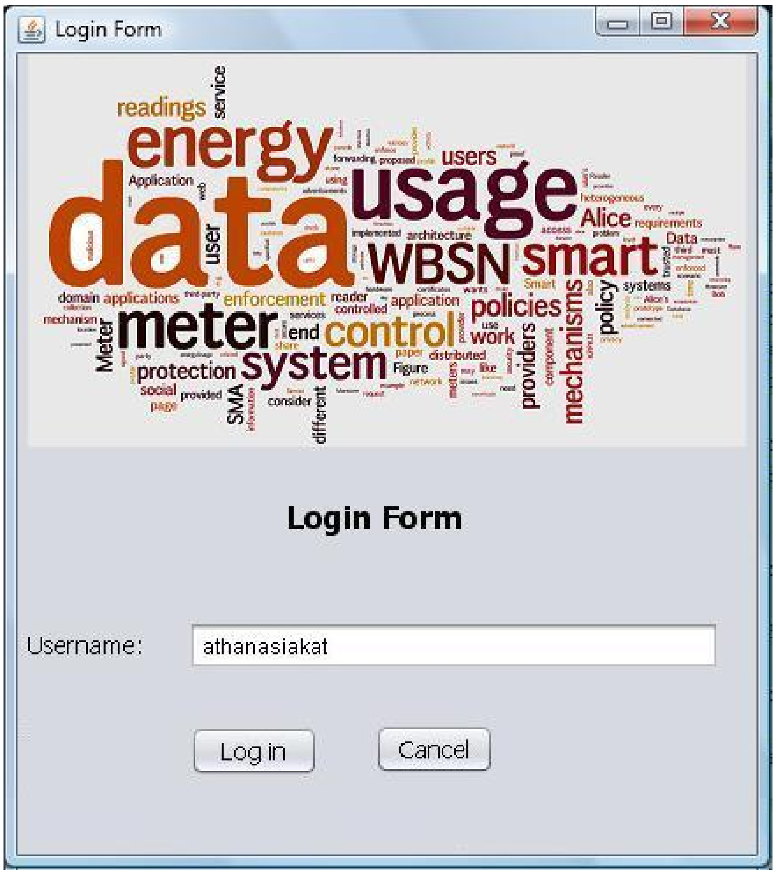


Figure 5.3: Login Page.

part) is called (see Figure 5.1) in order to enable the usage control of images in our application. The function *startUsing(String subject, String object, String mode)* is called. This function enables the UCON engine. The arguments of this function include the subject's and object's name and the mode that the subject has chosen to access the object ('load' or 'edit').

In order to check the preConditions, it calls the functions *checkPreConSubj(String subject, String object, String condType, String mode)*,*checkPreConObj(String subject, String object, String condType, String mode)*, and *checkPreConCon(String context)*. The first function checks the preConditions of a subject. It has the same arguments, as input, with the startUsing function. The second function checks the preConditions of an object. The third function checks the preConditions of the context. If preConditions are met successfully, then the function *checkForAccess(String subject, String object, String context, String mode)* is called, in order to check if the specific right ('load'/'edit') is allowed for the pair (subject, object). If so, then s/he will be redirected to the Image Viewer or Image Editor (see Figure 5.5, Figure 5.7) for the first time.

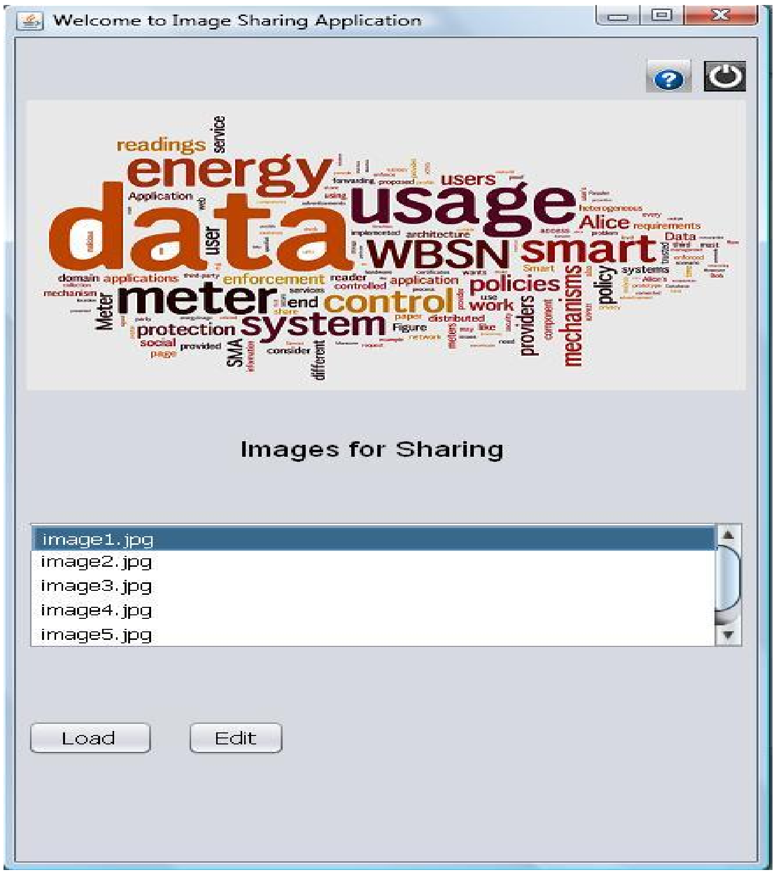


Figure 5.4: Welcome Page.

Then, the ongoingConditions are enforced in order to ensure the control during the access. The startUsing function calls the functions *checkOnConSubj(String subject, String object, String condType, String mode)* and *checkOnConObj(String subject, String object,String condType,String mode)*, in order to check the ongoingConditions of a subject and object, respectively. While the ongoingConditions are satisfied then the subject could access the object ('load'/'edit').

It is worth mentioning that both Image Viewer and Image Editor have been implemented and called from the part of UCON Engine (trusted part). A notification message may be displayed if the user has not got the permissions for the selected image (see Figure 5.6, Figure 5.8).

5.3 Scenarios

In this section, we present two scenarios that allow loading/editing an image. Let us assume that Athanasia (subject) works at Computer Science Department (subject attribute) and she is responsible for designing the UCON$_{ABC}$ model (subject attribute). She has been given a username athanasiakat (subject attribute) from her Department. Furthermore, her Department has a folder that contains images (objects) which can be viewed or edited under some specific conditions (preConditions or/and ongoingConditions) that have to be fulfilled.

Scenario # 1: Load an Image

Scenario 1: Athanasia (subject) has to belong to the Computer Science Department (preCon) and she has to be in France (contextual information) in order to have the permission for accessing (right : read) "image1.jpg" (object), 7 times (mutability) from this folder and each time she can access it for 5 seconds (onCon).

Athanasia is logged in the application by providing her username, as we described it above (see Figure 5.3), and directed to the list of images (see Figure 5.4). When Athanasia clicked on the image and then pressed the Load button, the UCON engine (ucon engine part) is enforced (see Figure 5.1) in order to enable the usage control of images and check if this user has the appropriate permissions for accessing ('load') this image.

The UCON Engine starts (*startUsing("athanasiakat", "image1.jpg","France", "load")*) by checking the preConditions for the subject (*checkPreConSubj("athanasia kat", "image1.jpg", "preCon", "load")*), the preConditions for the object (*checkPre ConObj("athanasiakat", "image1.jpg", "preCon", "load")*) and the preConditions for the context (*checkPreConCon("France")*). More precisely, in our case, the UCON Engine checks if athanasiakat belongs to Computer Science Department (see the Tables in Section 5.1.1) and if the access time is elapsed. It also checks if the location is France (see Table 5.9) by retrieving the IP address of athanasiakat and calling the function *returnCountry("12893 135222", "Athanasia-PC", stmt, String "fr")* that returns the name of the country.

If the preConditions are met, then it checks if she has permissions for exercising the load operation (right:'read') (*checkForAccess("athanasiakat", "image1.jpg", "load")*). If so, the attributes which are related to accessTime (accessTime = accessTime + 1), as well as the variable which represents the seconds of each access are changed (mutability), otherwise she is redirected to the list of images. Then, the ongoingConditions are enforced (*checkOnConSubj("athanasiakat", "image1.jpg", "load")*, *checkOnConObj("athanasiakat", "image1.jpg", "load")*), in order to check during the access if the conditions are met or not. If she has not the appropriate permission for loading the image or when conditions are not met anymore (i.e the available seconds for loading this image are elapsed) then she can not exercise any rights to the image (see Figure 5.6) and she is redirected to the list of images again.

Scenario # 2:Edit an Image

Scenario 2: Athanasia (subject) has to belong to the Computer Science Department (preCon) in order to have the permission for accessing (right : write) "image2.jpg"(object), 2 times (mutability) from this folder.

Athanasia is logged in the application by providing her username, as we described it above (see Figure 5.3), and directed to the list of images (see Figure 5.4. When Athanasia clicked on the image and then pressed the Edit button, the UCON engine (ucon engine part) is called (see Figure 5.1) in order to enable the usage control of images and checks if this user has the appropriate permissions for accessing ('edit') this image.

The UCON Engine starts (*startUsing("athanasiakat", "image2.jpg", "edit")*) by checking the preConditions for the subject and object (*checkPreConSubj("athana siakat", "image2.jpg","preCon", "edit")*, *checkPreConObj("athanasiakat", "image2. jpg", "preCon", "edit")*. More precisely, in our case, the UCON Engine checks if athanasiakat belongs to Computer Science Department (see the Tables in Section 5.1.1) and if the access time is elapsed.

If the preConditions are met, then it checks if she has permissions for exercising the load operation (right:'read') (*checkForAccess("athanasiakat", "image2.jpg", "edit")*). If so, the attribute which is related to accessTime (accessTime = accessTime + 1) (mutability) is changed, otherwise she is redirected to the list of images. Then, the ongoingConditions are enforced (*checkOnConSubj ("athanasiakat", "image2. jpg", "edit")*, *checkOnConObj("athanasiakat", "image2.jpg", "edit")*), in order to check during the access if the conditions are met or not. If she has not the appropriate permission for editing the image or when conditions are not met anymore then she can not exercise any rights to the image (see Figure 5.8) and she is redirected to the list of images again.

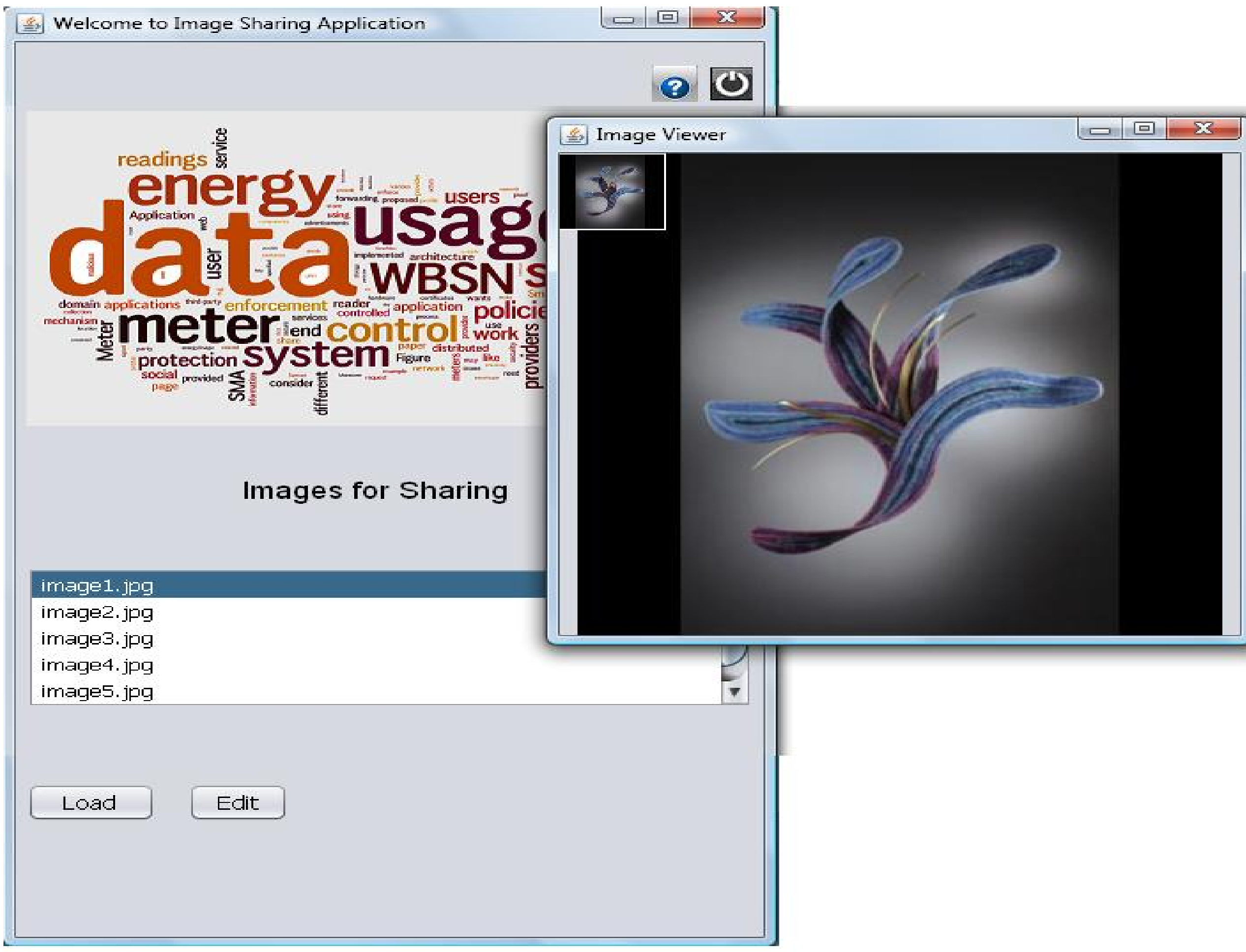
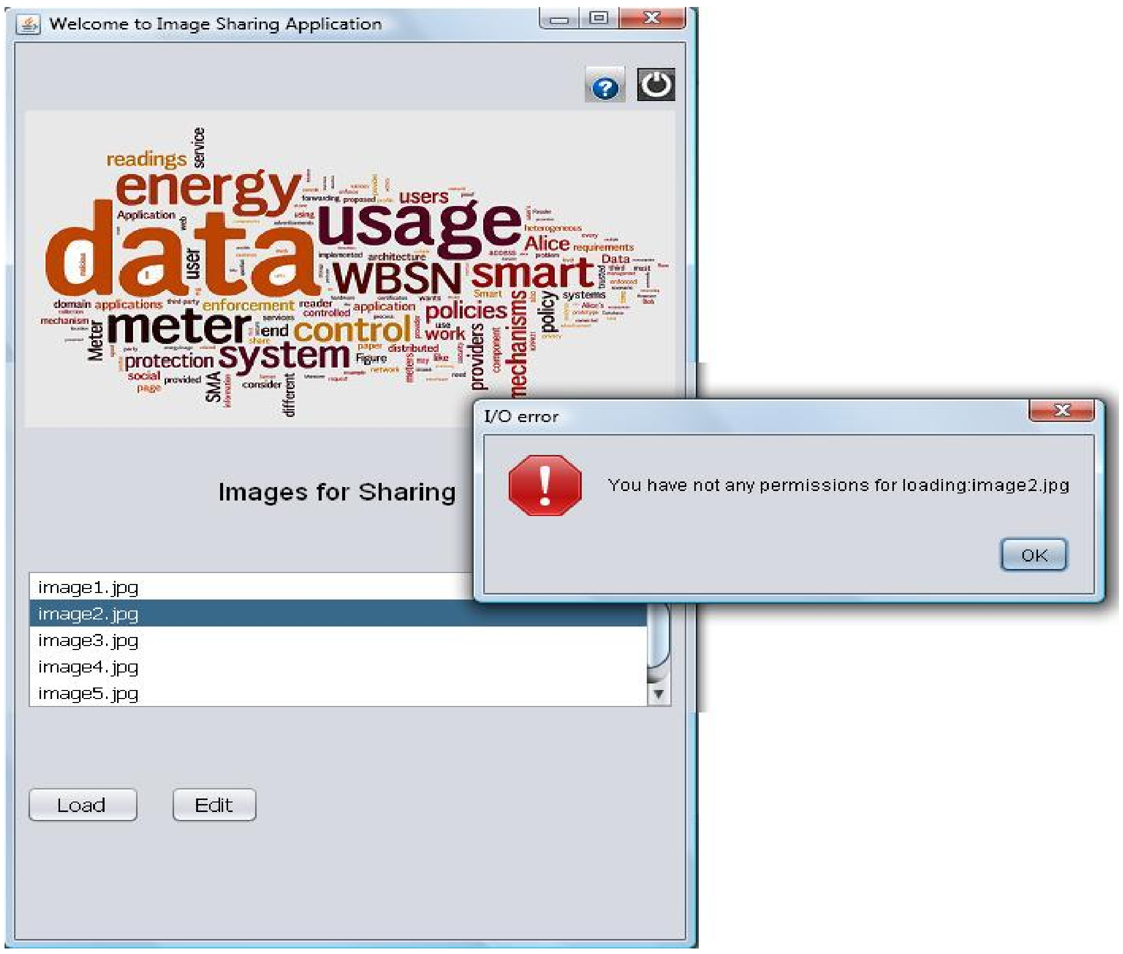


Figure 5.5: Load Image.



You have not any permissions for loading:image2.jpg

Figure 5.6: No permission for Loading an Image.

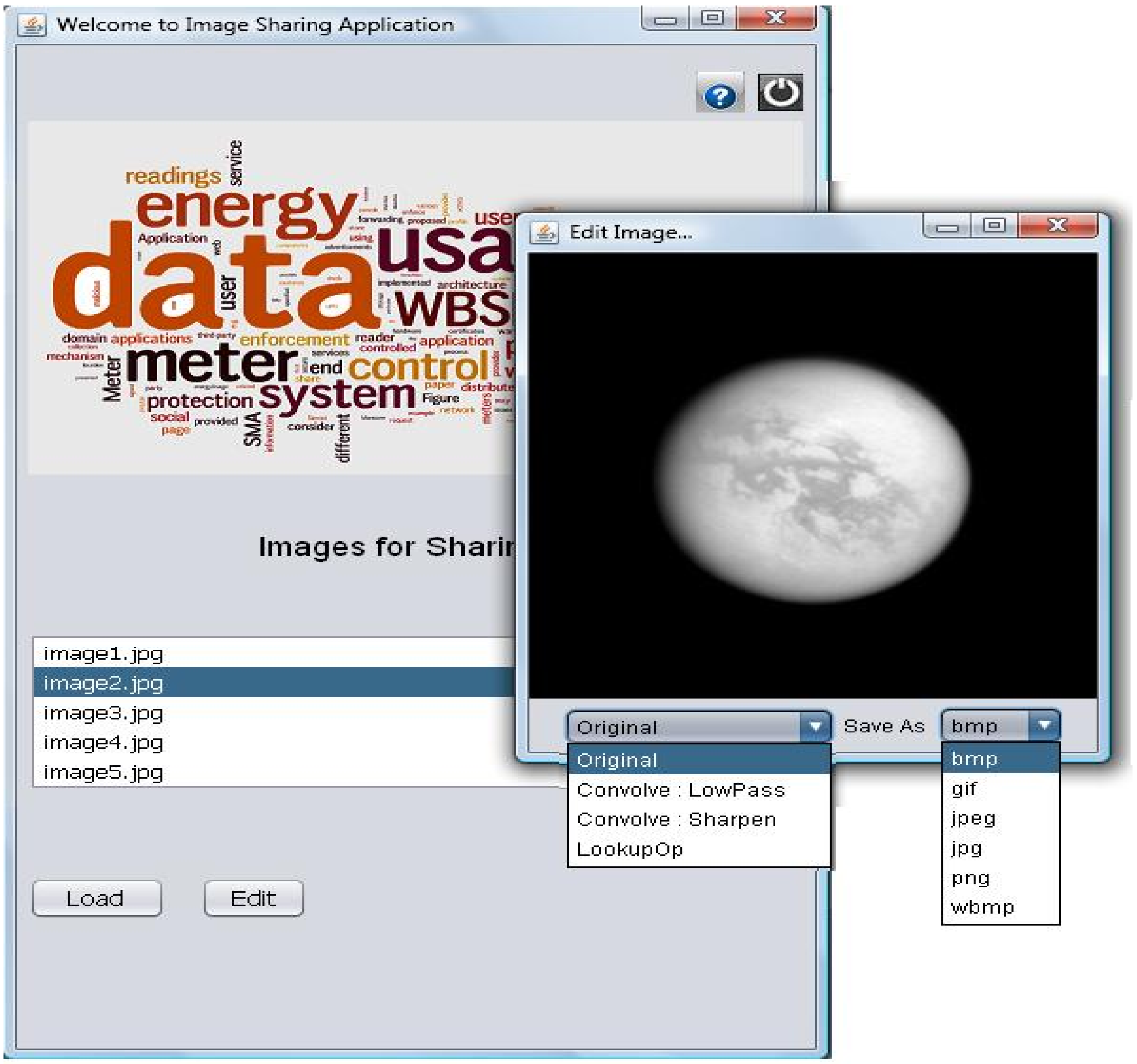
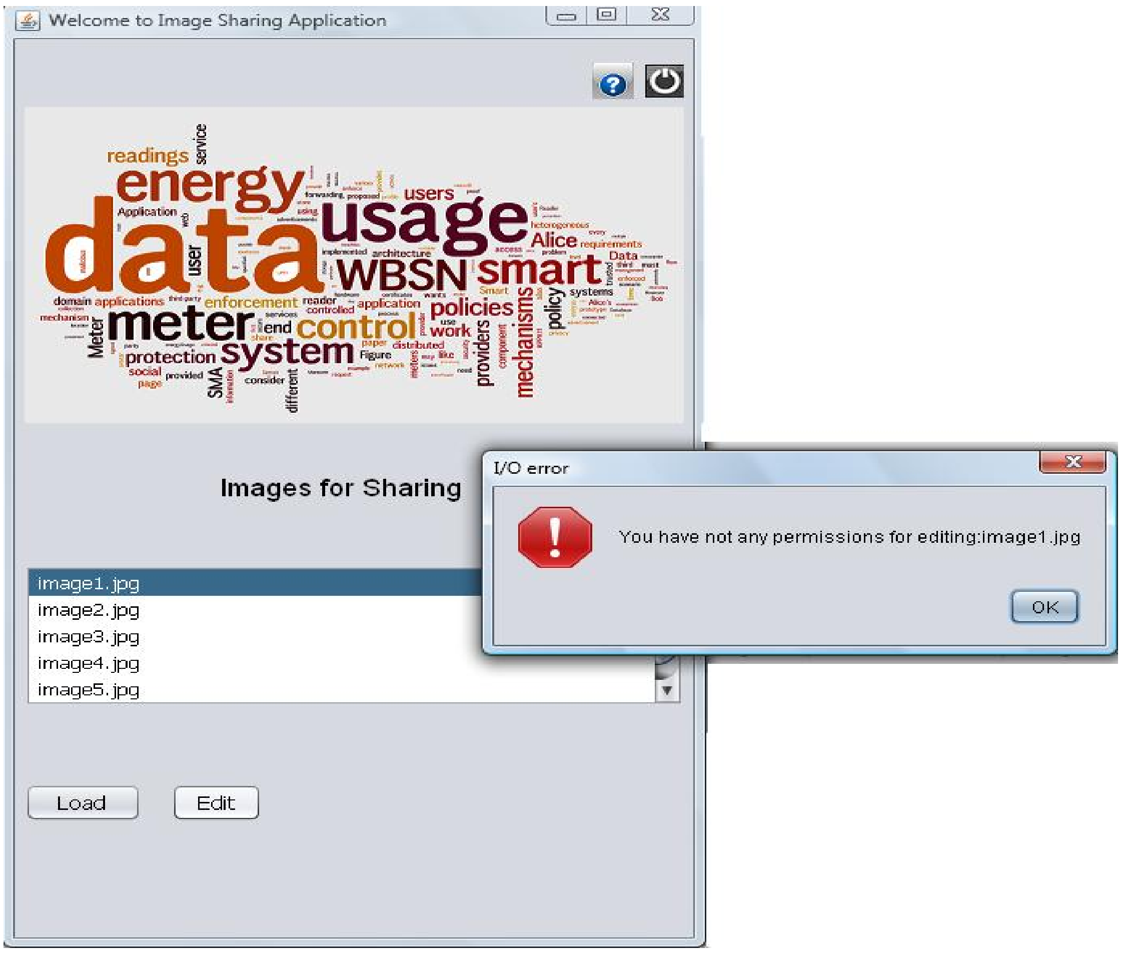


Figure 5.7: Edit Image.



You have not any permissions for editing:image1.jpg

Figure 5.8: No permission for Editing an Image.

Chapter 6

Conclusions, Discussion and Future Research

In this work, we discussed about Access Control (see Chapter 2) and Usage Control (see Chapter 3), demonstrating the state of the art on Access and Usage Control. More specifically, in Chapter 2, we gave some definitions on Access Control and described some Access Control models. We exhibited the limitations of these Access Control models. In Chapter 3, we exhibited some definitions on Usage Control and described in details the UCON$_{ABC}$ model, which is a powerful model and is able to cover the most of the limitations which are faced by Access Control.

In the last two Chapters (Chapter 4 and Chapter 5) we illustrated two implementations of the conceptual UCON$_{ABC}$ model. The former implementation can be a part of a system and its detailed description has been made in Chapter 4. Actually, it is an Engine that could be embedded within a system, proving the usage control of individuals data. This implementation has been intended to be used either into a social network application or as a service within a system like a reference monitor to handle the usage control requirements of the objects and mediate access to resources. The latter implementation consists of an application in which we have embedded the UCON Engine. In order to simplify the former implementation, in this part we used Data structures and a Database. A graphical user interface (GUI) has been designed as well. This application could run on any devices like PDSs [44, 45].

As we discussed in Chapter 4, as far as the former implementation is concerned, we developed the UCON model in java and we created the UCON policies in XML (XACML) [32, 33, 34, 35, 36, 37, 38]. The reason why we have chosen to implement this model in Java language instead of other languages (i.e. C, C++, Python) is that Java offers plenty of libraries. Some examples include the Xalan-Java which fully implements XSL Transformations (XSLT) [1] and the XML Path Language

1. XSLT is the first part of the XSL stylesheet language for XML and includes the XSL Transformation vocabulary and XPath.

(XPath)[2].

We have chosen to implement the Policies of this model using XML (XACML), taking advantages of XML. First of all, XML is very simple language and allows the developers to store the structured data into XML files. For instance, XML files can keep records instead of a Database. Those file could be changed easily when is needed. Second, almost all the technologies using XML file in order to store data in an open environment, such as the Internet. Third, XML is supported by PHP, Java, ASP, .NET, C, C++, Perl, Python and almost all programming languages.

In Chapter 5, we demonstrated a new perspective on the implementation of the UCON$_{ABC}$ model in our latter implementation using Data Structures (SQL) and a Database in order to extend and simplify the former implementation. More precisely, we designed a SQL schema with all the appropriate tables which constitute representations of the conceptual UCON$_{ABC}$ model [1, 2]. For the sake of simplicity, we have used the SQL that is very fast and efficient, especially if lots of I/Os have to take place. We realized that SQL worked well to get and store data. However, in the latter implementation, no history is recorded, as opposed to the former implementation.

In future work, considering the Personal Data Server (PDS) context [44, 45] and the short amount of memory in a PDS, we could use the latter implementation and we could imagine this application as a part of a Social Network, in order to protect individuals data. In this case, it could be embedded in a Personal Data Server, providing Secure Data Sharing. This Data Sharing Application constitutes a general engine for individuals data protection. By this application, after the evaluation of policies, a usage decision (i.e. granted, denied) has been made, providing security in the Data Sharing transactions.

To sum up, in view of the above considerations I believe that the Social Networking Research Area could be benefited from this work. We could imagine a safer Social Network which could be benefited from the UCON engine for the Data Sharing (i.e. images, other files), as a mean to share data safely.

2. XPath is a language for addressing parts of XML documents

Bibliography

[1] J. Park and R. Sandhu, “Towards usage control models: beyond traditional access control,” in *Proceedings of the seventh ACM symposium on Access control models and technologies*, ser. SACMAT ’02. New York, NY, USA: ACM, 2002, pp. 57–64. [Online]. Available: http://doi.acm.org/10.1145/507711.507722

[2] ——, “The uconabc usage control model,” *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 128–174, Feb. 2004. [Online]. Available: http://doi.acm.org/10.1145/984334.984339

[3] P. Samarati and S. D. C. d. Vimercati, “Access control: Policies, models, and mechanisms,” in *Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, ser. FOSAD ’00. London, UK, UK: Springer-Verlag, 2001, pp. 137–196. [Online]. Available: http://dl.acm.org/citation.cfm?id=646206.683112

[4] N. Li, “Discretionary access control,” in *Encyclopedia of Cryptography and Security (2nd Ed.)*, 2011, pp. 353–356.

[5] Z. Mao, N. Li, H. Chen, and X. Jiang, “Combining discretionary policy with mandatory information flow in operating systems,” *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 3, p. 24, 2011.

[6] N. C. S. C. (U.S.), *A Guide to understanding discretionary access control in trusted systems.* National Computer Security Center, 1987. [Online]. Available: http://books.google.fr/books?id=BTM-AAAAMAAJ

[7] B. W. Lampson, “Protection,” *SIGOPS Oper. Syst. Rev.*, vol. 8, no. 1, pp. 18–24, Jan. 1974. [Online]. Available: http://doi.acm.org/10.1145/775265.775268

[8] V. Hu, R. Kuhn, T. Xie, and J. Hwang, “Model checking for verification of mandatory access control models and properties,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 21, no. 1, pp. 103–127, 2011.

[9] Z. Shan, X. Wang, and T. cker Chiueh, "Enforcing mandatory access control in commodity os to disable malware," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 4, pp. 540–554, 2012.

[10] S. Osborn, "Mandatory access control and role-based access control revisited," in *PROCEEDINGS OF THE 2ND ACM WORKSHOP ON ROLE-BASED ACCESS CONTROL*. ACM Press, pp. 31–40.

[11] D. Ferraiolo, D. Kuhn, and R. Chandramouli, *Role-Based Access Control*, ser. Artech House Computer Security Series. Artech House, 2003. [Online]. Available: http://books.google.fr/books?id=48AeIhQLWckC

[12] Y. Beresnevichiene, "A role and context based security model," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-558, Jan. 2003. [Online]. Available: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-558.pdf

[13] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, Feb. 1996. [Online]. Available: http://dx.doi.org/10.1109/2.485845

[14] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proceedings of the 1996 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1996, pp. 164–173.

[15] E. Wobber and M. Burrows, "Authentication in the taos operating system," *ACM Transactions on Computer Systems*, vol. 12, pp. 256–269, 1994.

[16] C. M. Ellison, C. Inc, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen, "Spki certificate theory," *IETF RFC*, 1999.

[17] D. Balfanz, "A security infrastructure for distributed java applications," in *In 21th IEEE Computer Society Symposium on Research in Security and Privacy*, 2000, pp. 15–26.

[18] A. Herzberg, Y. Mass, J. Michaeli, D. Naor, and Y. Ravid, "Access control meets public key infrastructure," in *In Proceedings of the 2000 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 2000, pp. 2–14.

[19] S. Guth, "A sample drm system," in *Digital Rights Management*, 2003, pp. 150–161.

[20] Q. Liu, R. Safavi-Naini, and N. P. Sheppard, "Digital rights management for content distribution," in *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21*, ser. ACSW Frontiers '03. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003, pp. 49–58. [Online]. Available: http://dl.acm.org/citation.cfm?id=827987.827994

[21] N. Rump, "Definition, aspects, and overview," in *Digital Rights Management*, 2003, pp. 3–15.

[22] R. Gooch, "Requirements for drm systems," in *Digital Rights Management*, 2003, pp. 16–25.

[23] N. Paskin, "Identification and metadata," in *Digital Rights Management*, 2003, pp. 26–61.

[24] R. Lepro, "Cardea: Dynamic access control in distributed systems," Tech. Rep., 2003.

[25] E. Damiani, S. D. C. di Vimercati, and P. Samarati, "New paradigms for access control in open environments," in *SIGNAL PROCESSING AND INFORMATION TECHNOLOGY*, 2005, pp. 540–545.

[26] M. Sastry, R. Krishnan, and R. S, "A new modeling paradigm for dynamic authorization in multi-domain systems."

[27] M. J. Covington and M. R. Sastry, "A contextual attribute-based access control model," in *Proceedings of the 2006 international conference on On the Move to Meaningful Internet Systems: AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS- CIAO, MONET - Volume Part II*, ser. OTM'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 1996–2006. [Online]. Available: http://dx.doi.org/10.1007/11915072_108

[28] C. Grompanopoulos, A. Gouglidis, and I. Mavridis, "A use-based approach for enhancing ucon," 2012.

[29] E. A. C. M. L. (XACML). [Online]. Available: http://xml.coverpages.org/xacml.html

[30] X. L. Reference. Online: http://gryb.info/xacml/doc/xacmlightreference.html. [Online]. Available: http://gryb.info/xacml/doc/XACMLightReference.html/

[31] S. X. Implementation. [Online]. Available: http://sunxacml.sourceforge.net/

[32] B. DuCharme, *XML: the annotated specification*, ser. The Charles F. Goldfarb series on open information management. Upper Saddle River, NJ 07458, USA: Prentice-Hall PTR, 1999.

[33] X. Apache. [Online]. Available: xerces.apache.org/xerces-j/

[34] X. Resolver. [Online]. Available: docs.oracle.com/javaee/5/api/javax/xml/stream/XMLResolver.html

[35] X. Parser. [Online]. Available: stackoverflow.com/questions/373833/best-xml-parser-for-java

[36] X. reader. [Online]. Available: www.mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/

[37] J. A. for XML Processing. [Online]. Available: en.wikipedia.org/wiki/Java_API_for_XML_Processing

[38] OASIS-open. [Online]. Available: https://www.oasis-open.org/committees/download.php/2713/

[39] M. Matthews, J. Cole, and J. D. Gradecki, *MySQL and Java Developer's Guide*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[40] O. Documentation. [Online]. Available: http://docs.oracle.com/

[41] W. S. I. in SQL. [Online]. Available: http://www.w3schools.com/sql/sqlintro.asp

[42] Databases. [Online]. Available: http://www.cs.nott.ac.uk/~nza/G51DBS/dbs19.pdf

[43] Jdbl. [Online]. Available: http://jdbi.org/

[44] T. Allard, N. Anciaux, L. Bouganim, Y. Guo, L. L. Folgoc, B. Nguyen, P. Pucheral, I. Ray, I. Ray, and S. Yin, "Secure personal data servers: a vision paper," *PVLDB*, vol. 3, no. 1, pp. 25–35, 2010.

[45] N. Anciaux, L. Bouganim, Y. Guo, P. Pucheral, J.-J. Vandewalle, and S. Yin, "Pluggable Personal Data Servers." [Online]. Available: http://hal.inria.fr/inria-00551836