

Authors	E. Bertino E. Ferrari A. Perego	Organisation	UNIMI		
Title	Detailed description of the Milan model				
Publication Date	22/05/2001	Document Reference	Deliverable/D8	Task Reference	T4.1
Circulation (C,R,P)	R	Document Type	Deliverable	Document Version	1.0
Deliverable Reference	D8				

DELIVERABLE D8

Detailed description of the Milan model

Table of Contents

1	Introduction.....	3
2	The Framework of the Milan Model.....	3
3	Overview of the Milan Model.....	5
4	The Basic Components of the Milan Model.....	7
4.1	DL Objects	7
4.2	Credentials.....	10
4.2.1	The Credential Constraint Specification Language	12
4.2.2	Credential Specification.....	13
4.3	Privileges.....	14
5	The Milan Authorisation Model.....	15
5.1	Access Authorisations.....	15
5.2	Authorisation Propagation and Conflict Resolution.....	18
6	The Milan Model Access Control.....	21
7	The Architecture of the Milan Model.....	24
8	Conclusions	25
	References	26
	Appendix.....	27

1 Introduction

In this deliverable we supply a full description of the Milan model, one of the two main conceptual tools that will be used in the EUFORBIA project, and which has been briefly illustrated in Deliverable 2.

We recall here that the Milan model will be used in the filtering component of the EUFORBIA prototype, in parallel with direct utilisation of NKRL EUFORBIA labels. In this particular modality of filtering, whenever a user requests a Web page the Milan module receives as input concepts extracted from the EUFORBIA labels associated with the document and pertaining to the NKRL hierarchy of concepts (H_CLASS). The module then verifies, with respect to the authorisation (filtering) policies chosen by the user, whether that Web page can be displayed or not in the user's browser.

The remainder of this deliverable is organised as follows. Section 2 illustrates the general framework in which the Milan model was conceived and developed. Section 3 presents an overview of the Milan model approach. Section 4 introduces preliminary concepts on which the Milan model relies. Section 5 introduces the Milan authorisation model, whereas Section 6 deals with access control. Section 7 describes the prototype system implementing the Milan model. Section 8 concludes the deliverable. Finally, Appendix lists the symbols more frequently used in this deliverable and reports a brief explanation of their meaning.

Formal proofs are not included in the deliverable. We refer the reader to [1].

2 The Framework of the Milan Model

The Milan model is a content-based authorisation model for Digital Libraries. Digital Libraries (DL) are systems that carry out interactions among information and knowledge producers, librarians, and information and knowledge seekers [2]. DL's goal is to provide the ability on a global scale to acquire, store, and retrieve information electronically. DL systems are typically a collection of distributed, autonomous and heterogeneous sites. They deal with large amount of multimedia information where objects may be stored on a variety of formats and media such as disk, tape or CD-ROM and typically come from a variety of sources which may wish to control the DL use (retrieval or modification) or to add value to the content. Users of these systems have a wide variety of diverse background and interests and usually access the DL from remote sites.

One of the problem areas in DL is securing information. On one hand the system must be secure against malicious use and data-corruption and must ensure the privacy of its users as well as protect the intellectual property of the vendors and information producers. On the other hand, however, the system should provide open access so that vendors and information producers can add/update information and services any time. Thus, a full-fledged security model for DLs should address authentication and security services to guarantee privacy, integrity, confidentiality, access control, and copyright issues. In this deliverable we focus only on issues related to access control. We plan, however, to gradually extend our security model to cover the above-mentioned aspects.

In conventional database environments access control is usually performed against a set of authorisations stated by security administrators or users according to some security policies [8]. An

authorisation in general is specified on the basis of three parameters, $\langle s, o, p \rangle$. This triple specifies that user s is authorised to exercise privilege p on object o . Such an approach to authorisation specification is not suitable for a DL environment due to its highly dynamic user population and its extraordinarily large collection of objects. In this deliverable, we describe an authorisation model in which access policies can be based on user qualifications and characteristics (for example, a user can be given access to an R rated video only if he/she is older than 18 years) and the content of the object or part of an object (for example, all documents containing discussions on how to operate guns must be made available only to users who are 18 or older). The Milan model enables enforcement of security policies in conventional libraries, thus making the transition from paper based libraries to their digital counterparts easier. Moreover, the Milan model is able to support articulated access control measures. For example, in traditional libraries it is not always possible to give access only to a part of a book, whereas our access control model will support selective accesses to components of DL objects.

The relevance application domain for the Milan model is the Global Legal Information Network (GLIN), a project originally undertaken by the Law Library of Congress (LLOC). In the following, we present an example from GLIN that demonstrates the necessity of new security features for a DL environment.

Example 2.1 The GLIN project began in 1993 in response to its responsibility to support the information needs of the United States Congress, the United States Supreme Court and the executive agencies in the areas of foreign, international, and comparative law. The broad goal of GLIN is to create a knowledge base of international law and to make this knowledge base available to member countries from around the world (about 35 and growing). GLIN emerged from a long-standing experience in manually sorting, arranging, and indexing primary sources of law to gain controlled access to these sources. These are needed for analysis to prepare studies, reports, and memoranda in response to questions submitted to the LLOC. GLIN represents a unique attempt to create an environment where international transactions and interactions are based on trustworthy information and to expanding bonds among member countries that will help fuel growth of common markets and greater cultural ties.

In GLIN, only a certain group of users such as members of official government agencies or U.S. Congress can gain access to certain documents. For example, the Law Library of Congress publishes monthly a *World Law Bulletin*, which includes a *blue page report* that provides a report about pending legislation which is a particular topic of interest to Congress. This report should not be public until it is open for public discussion because this could be a gold mine to lobbyists.

Moreover, based on the content of the report, only certain individuals are allowed to view or generate this report or parts of it. For example, the report, "Attorney's fees and Litigation Expenses in Selected Foreign Nations," publishes an overview of attorneys' fees legislation in fifteen countries, which consists of fifteen parts, one for each country. In order to produce a part pertaining to a country, it is required that an individual, such as a legal research analyst, has the national origin of that country.

Although all the examples in this deliverable are from a GLIN environment, similar access policies can be found in other domains as well (for example only subscribers to a magazine are allowed to access that magazine).

Since traditional authorisation models are not capable of specifying or enforcing such policies, in this deliverable, we describe a content-based authorisation model suitable for a DL environment. The Milan model allows granting of privileges based on user qualifications and characteristics called *credentials* rather than users themselves, and based on the concepts associated with objects

rather than the objects themselves. Specifically, our authorisation model provides (1) flexible specification of authorisations based on the qualifications and characteristics of users (including positive and negative authorisations) rather than just on specific users; (2) access control specification to digital library objects that is based not only on object identifiers but also on objects content; and (3) varying granularity of authorisation objects ranging from sets of library objects to specific portions of objects. The resulting model is therefore very flexible and powerful in terms of the kind of protection requirements it can represent. However, the flexibility provided to the user makes access control more complex. The main reason is that users accessing a DL are usually global users (that is, they are not users belonging to the organisation owning the DL). Global accesses pose a number of problems, especially when global users are required to provide their credentials for access control purposes. Access control information that are recorded for users at the users' organisation may differ with respect to the credentials required by the access control system of the DL. Consider for instance the case of a DL requiring the age of the user for giving access to certain objects whereas the user's organisation keeps the birth date of the user. In such case it is easy to compute the required; however, more complicated situations may arise that require the establishment of some mappings among different sites. Another related question is how does a user (or an organisation) wishing to access a DL determine the information to supply for access control purposes. Finally, other important issues include the use of certification and other authentication mechanisms to ensure the authenticity of the information provided for access control purposes.

3 Overview of the Milan Model

Unlike traditional access control models where authorisations are specified as $\langle \text{user}, \text{object}, \text{privilege} \rangle$, in the Milan model authorisations are specified as $\langle \text{credentials}, \{ \text{concepts} \mid \text{label} \}, \text{privilege}, \text{sign} \rangle$.

Credentials represent the characteristics and qualifications of users. We associate *credentials* with each user. Such credentials are then used to determine the valid authorisations of that user based on *credential expressions*.

In the Milan model, content-based access control can be supported into two different and orthogonal ways:

1. by pre-processing the DL object using a document management mechanism (such as, for instance, in [3]), which is capable of extracting concepts from documents and building a conceptual hierarchy.
2. by using the labelling mechanism provided by PICS standard [5, 6].

A concept in a DL object means the abstractions or notions one expects to find in that object. In other words, a concept is a means for concisely describing the content of a DL object. However, concepts are not just keywords. For example, a combination of words in a document may elicit a sorrowful reaction from a reader although they do not contain any keywords related to "sadness". Often, concepts in a given domain (denoted as \mathcal{CP}) are interrelated, and therefore can be organised into a *conceptual hierarchy*, which is a partial order $\prec_{\mathcal{CP}}$,¹ where concepts towards the top of the hierarchy are more general and common to most DL objects, whereas concepts towards the bottom of the hierarchy are more specific. Given two concepts $cp_1, cp_2 \in \mathcal{CP}$, we say that cp_1 is a more specific concept than cp_2 if and only if $cp_1 \prec_{\mathcal{CP}} cp_2$. Figure 1 depicts the hierarchy of concepts typically encountered in GLIN. In the figure, there is an arc from concept cp_1 to concept cp_2 if cp_2

¹ By hierarchy we mean a poset (S, \prec) where S is a set and \prec is a partial order over S .

$\leq_{CP} cp_1$. Thus, concepts towards the top of the hierarchy are more general and common to most DL objects, whereas concepts towards the bottom of the hierarchy are more specific. For instance, the concept `Imports Tax` can be regarded as a specialisation of the concept `Import-Export`.

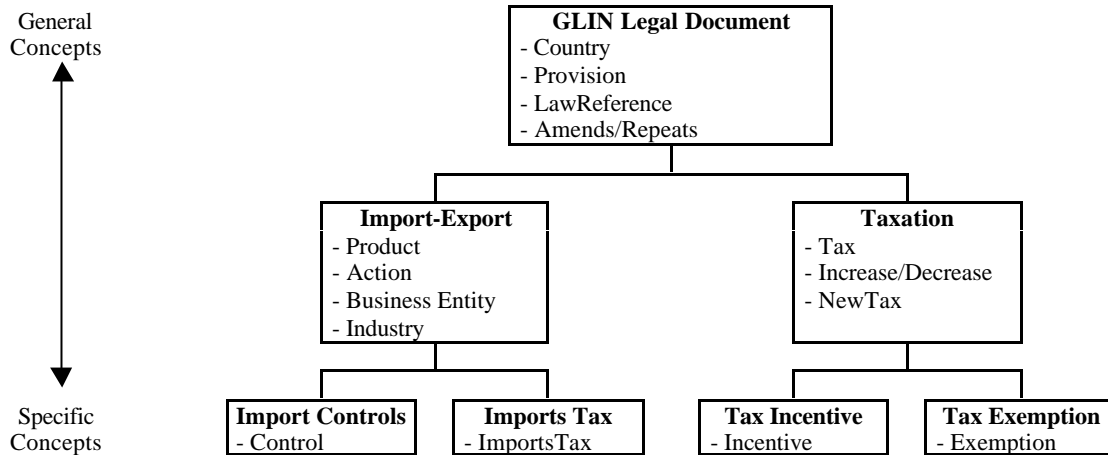


Figure 1: An example of conceptual hierarchy

PICS, Platform for Internet Content Selection [5, 6], establishes Internet conventions for label formats and distribution methods. The PICS conventions have caught on quickly. In early 1996, Microsoft, Netscape, SurfWatch and other software vendors announced PICS-compatible products. The basic idea of PICS is that not everyone needs to block reception of the same materials. Parents may not wish to expose their children to sexual or violent images. Businesses may want to prevent their employees from visiting recreational sites during hours of peak network usage. Governments may want to restrict reception of materials that are legal in other countries but not in their own. The “off” button (or disconnecting from the entire Net) is too crude: there should be some way to block only the inappropriate material.

The basic idea of the PICS standard is to interpose selection software between the recipient and the on-line documents. The software checks labels to determine whether to permit access to particular materials. It may permit access for some users but not others, or at some times but not others.

A *content label* (or *rating*) contains information about a document. A content label has three parts:

1. the URL chosen by the rating service² as its unique identifier;
2. options which give additional properties of the document being rated as well as properties of the rating itself, such as the time the document was rated;
3. a set of attribute-value pairs that describe a document along one or more dimensions.

One or more labels may be distributed together as a list. Figure 2 illustrate the general form for a label list.

² A *rating service* is an individual, group, organisation, or company that provides content labels for information on the Internet. The labels it provides are based on a *rating system*. Each rating service picks a URL as its unique identifier. It is included in all content labels the service produces, to identify their source. A rating system specifies the dimensions used for labelling, the scale of allowable values on each dimension, and a description of the criteria used in assigning values.

```
(PICS-1.1
  <service-url> [option...]
  labels [option...] ratings (<category> <value> ... )
    [option...] ratings (<category> <value> ... )
    ...
  <service-url> [option...]
  labels [option...] ratings (<category> <value> ... )
    [option...] ratings (<category> <value> ... )
    ...
...)
```

Figure 2: General form for a label list

Content labels, as defined in the PICS standard, can be regarded as a set of pairs (*category*, *value*), where *category* is a concept describing the content of the document; and *value* is a numeric value which quantifies the “degree” of *category* in the related document. The set of categories is usually a flat domain. For example, the RSACi rating system — that is now fully integrated in the Microsoft Explorer and the Netscape Navigator — includes four content categories : Nudity, Sex, Language and Violence; each of them can be associated with numeric values, from 0 to 4, which correspond to particular “degrees” according to which this particular category is present in the document.

The Milan model supports *browsing* and *authoring* privileges with various subtypes within each privilege type. These privilege types subsume the conventional privileges such as read and write. Additionally, we provide support for the specification of negative as well as positive authorisations.

Thus, our authorisation model provides both content-dependent and content-independent access control to DL objects. Moreover, in the Milan model, authorisations can be specified on the basis of either the user identity or user characteristics.

4 The Basic Components of the Milan Model

In this section, we introduce the basic components on which our access control model relies. We first characterise how DL objects are represented in the Milan model. Then, we introduce the concept of *credential*. Finally, we illustrate the access privileges supported by the Milan model.

4.1 DL Objects

Each DL document, referred to as *digital library object (dlo)*, typically contains unstructured information of different media types (e.g., text, images, videos). We assume that each *dlo* is associated with a unique object identifier (ID) which is assigned by the system upon the object creation and remains unchanged for the life of the object.

Often different portions of the same *dlo* have varying protection requirements. As an example, consider the case of an article from a journal, where its abstract could be made available to everyone, whereas the rest of the article should be made available only upon subscription to the journal. For this reason, we allow the *dlo* to be divided into *slots*. A slot is a piece of information contained in a document that can be identified by a name. Like DL objects, slots are untyped in that they may contain different types of data. For example, *abstract*, *title*, *authors* and *bibliography* are slots that can be associated with a scientific article. Not all slots must be identified by a name. We consider a default unnamed slot as the slot containing the part of the *dlo* that is not included in any defined slot. If no slot of an object is identified by a name, the unnamed slot will be the entire object.

A *dlo* can also contain *links* to other related *dlos*. The semantics of a link connecting two *dlos* is that the contents of the two objects are related. For instance, a *dlo* representing a scientific article may contain links to publications on the same topic. Each link is represented as a triple (*link_id*, *source_id*, *destination_id*), where *link_id* is the link identifier and *source_id*, *destination_id* are the IDs of the two objects connected by the link.

In the following, we use *OI*, *LI*, *SN*, *CP* and *LB* to denote a set of DL object identifiers, a set of link identifiers, a set of slot names, a set of concepts, and a set of labels, respectively. Moreover, given a tuple *t* with component names c_1, \dots, c_n we use notation $c_i(t)$ to denote the value of component c_i , $i = 1, \dots, n$.

A *dlo* can therefore be represented as a 4-tuple (i , *slots*, *links*, {*concepts* | *label*}), where $i \in OI$ is the ID of the *dlo*; *slots* $\in 2^{SN}$ is a set of slot names identifying relevant portions within the *dlo*, *links* $\in 2^{LI}$ is a set of link identifiers, *concepts* $\in 2^{CP}$ is the set of relevant concepts in the *dlo*, and *label* $\in 2^{LB}$ is a set of pairs (*category*, *value*), as defined in the RSACi rating system, where *category* $\in \{\text{nudity, sex, language, violence}\}$ and *value* $\in \{0, \dots, 4\}$.

The following is an example of object definition with respect to the conceptual hierarchy of Figure 1.

Example 4.1:

- (O_1 , {America, Europe}, {link₁, link₂}, {Import Controls, Imports Tax, Taxation});
- (O_2 , {America, Europe}, {link₁, link₂}, {(language, 0), (violence, 0)}).

The *Object Base*, denoted by *OB*, is the set of all *dlos*. Given a $dlo = (i, slots, links, \{concepts \mid label\})$, we use $C(dlo)$ to denote the set of concepts characterising *dlo*, and $Cat(dlo)$ to denote the set of label categories characterising *dlo*. $C(dlo)$ is union of the sets of concepts in *concepts* and all the concepts more general than those in *concepts*. Formally, $C(dlo) = concepts \cup \{cp \mid cp \in CP \text{ such that } \exists cp' \in concepts, \text{ with } cp' <_{CP} cp\}$. Note that the definition of $C(dlo)$ is directly implied by the semantics of the $<_{CP}$ relationship in that, if the content of an object is described by a given concept, then it is also described by all the (more general) concepts which precede it in the conceptual hierarchy.

Access authorisations can be given either on specific objects (by listing their IDs) or on all the objects containing a particular concept (or a particular combination of concepts) or on all the objects whose labels contain a given category with a particular value. Moreover, a finer-grained access control can be enforced by authorising users to access only specific slots and/or links within an object.

All these possibilities are summarised by the notion of *entity specification*. Before formally introducing the notion of entity specification, we need to introduce the following definition.

Definition 4.1 (Conceptual Expression) The set \mathcal{COEX} of conceptual expressions is defined as follows:

- each element of \mathcal{CP} is a conceptual expression;
- if ce_1 and ce_2 are conceptual expressions, then $(ce_1 \wedge ce_2)$ and $(ce_1 \vee ce_2)$ are conceptual expressions.

Example 4.2 The following are examples of conceptual expressions:

- `Import-Export` refers to objects containing the concept `Import-Export`;
- `Import-Export` \vee `Tax Exemption` refers to objects containing the concept `Import-Export` or the concept `Tax Exemption`;
- `Import-Export` \wedge `Tax Exemption` refers to objects containing the concept `Import-Export` and the concept `Tax Exemption`.

Moreover, objects to which an authorisation applies can be defined by means of a *label condition* expressed by means of the specification language introduced in what follows.

Definition 4.2 (Label Condition) The set \mathcal{LC} of label conditions is defined as follows:

- if $cat \in \{\text{nudity, sex, language, violence}\}$, $v \in \{0, \dots, 4\}$ and $OP \in \{>, <, \leq, \geq, \neq, =\}$, then $cat \text{ OP } v$ is a label condition;
- if lc_1 and lc_2 are label conditions, then $(lc_1 \wedge lc_2)$ and $(lc_1 \vee lc_2)$ are label conditions.

Example 4.3 The following are examples of expressions in \mathcal{LC} :

- `violence > 2` refers to objects where the category `violence` is labelled with a value greater than 2;
- `violence = 3` \vee `language < 4` refers to objects where the category `violence` is labelled with a value equal to 3 or the category `language` is labelled with a value less than 4;
- `nudity` \geq 4 \wedge `language = 3` refers to objects where the category `nudity` is labelled with a value greater than or equal to 4 and the category `language` is labelled with a value equal to 3.

Thus, the notion of entity specification can be defined as follows.

Definition 4.3 (Entity Specification) An entity specification has one of the following two forms:

1. $\{\text{co-spec} \mid \text{label-spec}\}.\text{slot-spec}$, where `co-spec` is either a conceptual expression in \mathcal{COEX} , or a set, possibly empty, of object identifiers in 2^{OI} , `label-spec` is a label condition in \mathcal{LC} that can also be omitted, and `slot-spec` $\in 2^{SN}$ is a set, possibly empty, of slot names, or
2. `link-spec`, where `link-spec` $\in 2^{LI}$ is a set, possibly empty, of link identifiers.

The entity specification `co-spec.slot-spec`, denotes the slots whose names are in `slot-spec` of the objects denoted by `co-spec`. If `slot-spec` = \emptyset , the specification denotes all objects denoted by `co-spec`, whereas if `co-spec` = \emptyset , it denotes all slots whose names are in `slot-spec`, regardless of the objects in which they appear. If `co-spec` $\neq \emptyset$, the specification denotes all the objects whose concepts satisfy `co-spec`.

The entity specification `label-spec.slot-spec`, denotes the slots whose names are in `slot-spec` of the objects denoted by `label-spec`. If `slot-spec` = \emptyset , the specification denotes all objects denoted by `label-spec`, whereas if `label-spec` = \emptyset , it denotes all slots

whose names are in `slot-spec`, regardless of the objects in which they appear. If `label-spec` $\neq \emptyset$, the specification denotes all the objects whose concepts satisfy `label-spec`.

Similarly, `link-spec` denotes the links whose identifiers are in `link-spec`.

In practice, $\{\text{co-spec} \mid \text{label-spec}\}$, `slot-spec`, `link-spec` are omitted when they are empty, whereas they are represented by their unique element when they are a singleton. Moreover, given an entity specification `ent-spec` we use `co-spec(ent-spec)` to denote the set of object identifiers or the conceptual expression in `ent-spec`, `label-spec(ent-spec)` to denote the label condition in `ent-spec`, `slot-spec(ent-spec)` to denote the set of slot names in `ent-spec`, and `link-spec(ent-spec)` to denote the set of link identifiers in `ent-spec`.

Example 4.4 The following are examples of entity specifications:

- `(Import-Export \vee Tax Exemption)`, denotes all the objects containing the concept `Import-Export` or the concept `Tax Exemption`;
- `(Import-Export).{Introduction, America}` denotes slots `America` and `Introduction` of all the objects containing the concept `Import-Export`.
- `(language > 3 \vee violence \neq 4)`, denotes all the objects containing the category `language` labelled with a value greater than 3 or the category `violence` labelled with a value different from 4;
- `(nudity > 2).{Introduction, America}` denotes slots `America` and `Introduction` of all the objects where the category `nudity` is labelled with a value greater than 2.

Given a conceptual expression `cx`, `Concepts(cx)` indicates the set of concepts appearing in `cx`, whereas `Obj_c(cx)` denotes the set of IDs of all the `dlos` whose associated concepts (that is, those in `C(dlo)`) satisfy `cx`.

Similarly, given a label condition `lc`, `Categories(lc)` indicates the set of label categories appearing in `lc`, whereas `Obj_l(lc)` denotes the set of IDs of all the `dlos` whose associated label categories (that is, those in `Cat(dlo)`) satisfy `lc`.

4.2 Credentials

To allow for the specification of authorisations based not only on the user identity but also on the user characteristics, each user is associated with one or more *credential*. A credential is a set of user attributes that are needed for security purposes. Credentials are assigned when a user is created and are updated automatically according to the user's profile. To make the task of credential specifications easier, credentials with similar structures are grouped into *credential-types*.

To formalise the concepts of credentials and credential-types, we use \mathcal{AN} to denote a set of attribute names, \mathcal{T} to denote the set of possible types of attributes in \mathcal{AN} , and \mathcal{V} to denote the set of legal values for types in \mathcal{T} . \mathcal{T} includes both basic types (such as `integer`, `real`, `boolean`, `character`, and `string`) and structured types, built by applying the constructors, sets, lists, and records to basic as well as to structured types. Moreover, we denote the set of credential-type identifiers with \mathcal{CT} and the set of credential identifiers with \mathcal{CI} . We use U to denote a set of account identifiers, associated with the users authorised to access the system. Credential-types are formally defined as follows.

Definition 4.4 (Credential-type) A credential-type is a pair $(ct_id, attr)$, where $ct_id \in \mathcal{CT}$ is the credential-type identifier; and $attr$ is a set containing an item for each attribute of the credential type. $attr$ in turn is a triple (a_name, a_dom, a_type) , where $a_name \in \mathcal{AN}$ is the attribute name (the attribute names must be distinct), $a_dom \in \mathcal{T}$ is the attribute domain, and $a_type \in \{\text{opt}, \text{mand}\}$ indicates whether the attribute can assume a null value ($a_type = \text{“opt”}$) or not ($a_type = \text{“mand”}$).

Example 4.5 The following is an example of credential-type:

$(\text{employee}, \{(\text{age}, \text{string}, \text{opt}), (\text{address}, \text{string}, \text{mand}), (\text{salary}, \text{integer}, \text{opt}), (\text{nationality}, \text{string}, \text{mand}), (\text{national origin}, \text{string}, \text{mand})\})$.

Credential types in \mathcal{CT} are organised into a hierarchy, referred to as *credential-type hierarchy*, according to a partial order $<_{\mathcal{CT}}$. Given, two credential types ct_1 and $ct_2 \in \mathcal{CT}$ we say that ct_2 is a *subcredential type* (or shortly subtype) of ct_1 if and only if $ct_2 <_{\mathcal{CT}} ct_1$. Multiple inheritance is not supported, that is, we do not allow a credential-type to be a subtype of more than one credential-type. Each credential-type contains all the attributes of the credential-types preceding it in the hierarchy. Moreover, it can contain additional attributes, apart from the inherited ones. Given a credential-type $ct \in \mathcal{CT}$, $A(ct)$ denotes the set of attributes of the credential-type instances.

Example 4.6 Consider the credential-type *employee* of Example 4.5 and suppose that a new credential-type *legal research analyst*, subtype of *employee*, is created with an additional attribute *project*. Thus, $A(\text{legal research analyst}) = A(\text{employee}) \cup \{\text{project}\} = \{\text{age}, \text{address}, \text{salary}, \text{project}, \text{nationality}, \text{national origin}\}$, since *legal research analyst* inherits all the attributes of the credential-type *employee*.

An example of credential-type hierarchy is reported in Figure 3. In the figure, there is an arc from a credential-type ct_1 to a credential-type ct_2 if $ct_2 <_{\mathcal{CT}} ct_1$.

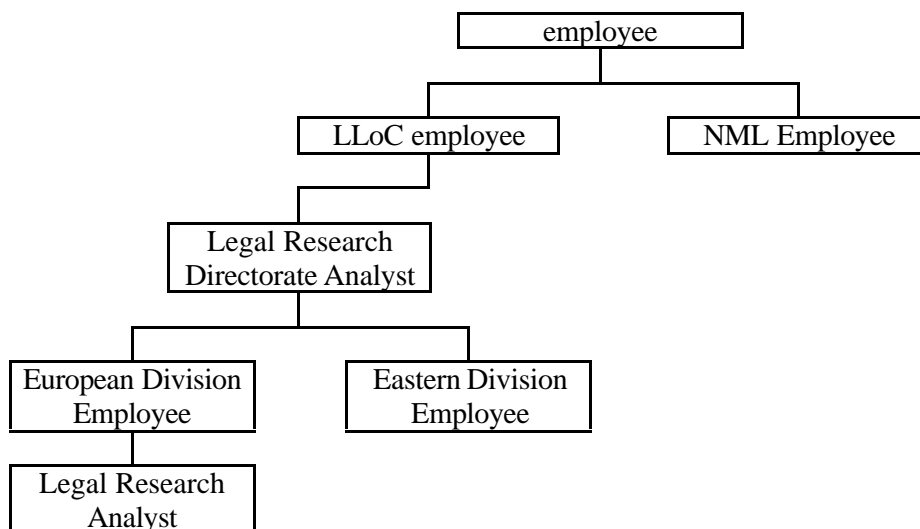


Figure 3: An example of credential-type hierarchy

Note that the existence of a common root of the entire credential-type hierarchy is not assumed. This means that the Security Administrator can structure the credential-type hierarchy into multiple direct trees with a distinguished root. Roots are classes without superclasses. However, we assume the existence of a built-in credential-type, denoted \top_{CT} , such that $ct <_{CT} \top_{CT}$, for all $ct \in CT \setminus \{\top_{CT}\}$. This predicate, which is automatically inserted by the system upon the creation of the credential type hierarchy, is used in the evaluation of authorisations not tagged to a specific credential-type, like for instance an authorisation for all the users older than 18.

A credential is an instance of a credential-type ct and is in turn a member of all credential-types ct' , such that $ct <_{CT} ct'$. (Here, we use the terminology adopted by object-oriented data models). A credential can be formally defined as follows.

Definition 4.5 (Credential) A credential c is a 4-tuple $(c_id, user_id, state, ct_id)$, where $c_id \in CI$ is the credential identifier; $user_id \in U$ is the identifier of the user with whom the credential is associated; $state = (a_1 : v_1, \dots, a_n : v_n)$, where $a_1, \dots, a_n \in A(ct_id)$ are the names of the attributes of c , $v_1, \dots, v_n \in \mathcal{V}$ are their values, and $ct_id \in CT$ is the identifier of the credential-type of which c is an instance.

The *Credential Base*, denoted CB , is the set of credentials associated with users in the system. Given a user u , $ct(u)$ denotes the set of credential types characterising user u . This set is composed by the credential types of which the credentials associated with u are either members or instances. Formally, let $ct' = \{ct \mid \exists (c_id, u', state, ct') \in CB \text{ such that } u = u' \text{ and } ct = ct'\}$. Then, $ct(u) = CT \cup \{ct \mid ct \in CT \text{ and } \exists ct' \in CT \text{ with } ct' <_{CT} ct\}$.

Example 4.7 The following are examples of credentials:

```
(c1, Bob, (age:null, address:Queen Street, salary:2,000, nationality:US,  
national origin:Italy), employee);  
(c2, Ann, (age:29, address:Broad Street, salary:null, project:P125,  
nationality:US, national origin:US), legal research analyst).
```

In the Milan model, authorisations can be given explicitly to users, by specifying their identifiers, or implicitly by imposing a set of conditions that the credential must satisfy. For instance, it is possible to specify that a particular object is accessible only to users whose age is greater than 18, or only to the legal research analysts working on a particular project. Conditions on credentials are expressed by means of the specification language we present in the next subsection.

4.2.1 The Credential Constraint Specification Language

The language the Milan model provides to express constraints on credentials consists of the following components:

- a set of variables Var_U , ranging over the set U of user identifiers;
- a set of predicate symbols $Pred$ of arity one, with type Var_U . For each credential-type $ct \in CT$, a corresponding predicate symbol $ct()$ is defined in $Pred$. Intuitively, predicates in $Pred$ are used to capture the associations of users with credential types.

Expressions that can be specified in our language, referred in the following as *credential expressions*, are formally defined as follows.

Definition 4.6 (Credential Expression) The set \mathcal{CE} of credential expressions is defined as follows:

- if $X \in \text{Var}_U$, $p \in \text{Pred}$, then $p(X)$ is a credential expression;
- if $X \in \text{Var}_U$, $a \in \mathcal{AN}$, $v \in \mathcal{V}$, and $\text{OP} \in \{=, \neq, >, <, \geq, \leq, \in, \notin, \subseteq, \not\subseteq, \subset, \not\subset, \supset, \not\supset, \ni, \not\ni\}$, then $X.a \text{ OP } v$ is a credential expression;³
- if ce_1 and ce_2 are credential expressions, then $(ce_1 \wedge ce_2)$ and $(ce_1 \vee ce_2)$ are credential expressions.

If a credential expression does not contain any credential-type, then its variables are implicitly bound to $\top_{\mathcal{CT}}$, that is, to the most general credential-type. Given a credential expression ce , $C_types(ce)$ denotes the set of credential types in ce . If ce does not contain any credential type, then $C_types(ce) = \{\top_{\mathcal{CT}}\}$.

Example 4.8 The following are examples of credential expressions in ce :

- `legal_research_analyst(X)`: intuitively this expression denotes the users that have a credential of type `legal_research_analyst` associated with them.
- `(X.project \neq p1)`: this expression denotes the users not involved in project `p1`.
- `(employee(X) \wedge X.age > 18)`: this expression denotes the employees whose age is greater than 18;
- `(Eastern_division_employee(X) \vee NLM_employee(X))`: this expression denotes the users with a credential of type `Eastern_division_employee` or `NLM_employee`.

4.2.2 Credential Specification

The users to which an authorisation applies can thus be identified by means of a *credential specification*, defined as follows.

Definition 4.7 (Credential Specification) A credential specification is either a set of user identifiers in 2^U , or a credential expression in \mathcal{CE} .

Each credential specification denotes a set, possibly empty, of users. For access control purpose, it is also important to identify the set of users that do not satisfy a given credential specification because of null values in their credentials. For instance, let $ce = X.salary > 2,000$. Suppose there exist two users u_1 and u_2 such that the value of attribute `salary` in u_1 's credential is 1,000, whereas the value of attribute `salary` in u_2 's credential is null. Thus, both u_1 and u_2 do not belong to the set of users identified by ce even if the value of u_2 's salary could be greater than 2,000. To keep track of these users, we introduce two functions, *Denotes* and *Undef* that, given a credential expression ce , return respectively the set of users identified by ce and the set of users that do not satisfy ce because of null values in their credentials. Table 1 lists, for each different type of credential expression that can be formulated in our language, the value that functions *Denotes* and *Undef* assume. With reference to the table, ce_1 and ce_2 are elements in \mathcal{CE} , $p \in \text{Pred}$, $X \in \text{Var}_U$, $a \in \mathcal{AN}$, $\text{OP} \in \{=, \neq, >, <, \geq, \leq, \in, \notin, \subseteq, \not\subseteq, \subset, \not\subset, \supset, \not\supset, \ni, \not\ni\}$, $v' \in \mathcal{V}$, $a(v)$ denotes the value of an attribute a .

³ Note that, the operators which can be used in an expression of the form $X.a \text{ OP } v$ are actually a subset of the ones listed above, since they depend on the type of the attribute $X.a$. We assume that there is some mechanism in place that verify that only the meaningful operators are used.

Credential Expression	Denotes()	Undef()
$p(X)$	$\{u \mid (c_id, user_id, state, ct_id) \in CB$ $user_id = u \text{ and } (ct_id = p \text{ or } ct_id <_{CT} p)\}$	\emptyset
$X.a \text{ OP } v'$	$\{u \mid (c_id, user_id, state, ct_id) \in CB$ $user_id = u, a \in A(ct_id), a(v) \text{ OP } v' \text{ holds}\}$	$\{u \mid (c_id, user_id, state, ct_id) \in CB$ $user_id = u, a \in A(ct_id), a(v) =$ $null\}$
$ce_1 \wedge ce_2$	$Denotes(ce_1)Denotes(ce_2)$	$Undef(ce_1) \cup Undef(ce_2)$
$ce_1 \vee ce_2$	$Denotes(ce_1)Denotes(ce_2)$	$Undef(ce_1) \cap Undef(ce_2)$
$\neg ce_1$	$U \setminus Denotes(ce_1)$	$Undef(ce_1)$

Table 1: Functions *Denotes()* and *Undef()*

Note that, according to the conditions listed in Table 1, the expression $p(X)$, where $p \in Pred$, denotes the users that have an associated credential whose type is either p or a subtype of p . For instance, with reference to Figure 3, the expression $(\text{European division employee}(X) \wedge X.age > 18)$, denotes all the users whose security type is either European division employee or legal research analyst and whose age is greater than 18.

Example 4.9 Let $CB = \{c_1, c_2\}$, where c_1, c_2 are the credentials in Example 4.7. Let $U = \{\text{Ann}, \text{Bob}\}$. $Denotes(\text{employee}(X)) = \{\text{Bob}, \text{Ann}\}$; $Denotes(X.age > 18) = \{\text{Ann}\}$; $Denotes((\text{employee}(X) \wedge \text{salary} \geq 2,000)) = \{\text{Bob}\}$; $Undef(\text{employee}(X)) = \emptyset$; $Undef(X.age > 18) = \{\text{Bob}\}$; $Undef((\text{employee}(X) \wedge \text{salary} \geq 2,000)) = \{\text{Ann}\}$.

4.3 Privileges

The set P of privileges supported by the Milan model is divided into two groups: *browsing privileges* and *authoring privileges*.

Browsing privileges allow users to see the information in an object (or in some of its slots) and/or to see the existence of a link in an object. Therefore, three different types of browsing privileges are supported: *view*, *link* and *view-all*. The *view* privilege allows a user to see the information in an object or in some of its slots. The *link* privilege authorises a user to see the existence of a specific link or of all the links in a given object. Clearly, for a *link* authorisation to be applicable, the user needs to be authorised to access the object source of the link, that is, the user must hold a *view* authorisation on the object. Finally, the *view-all* privilege subsumes both the *link* and the *view* privilege, since it allows users to see the information in an object and all the links it contains. Thus, the *view-all* privilege on an object is equivalent to the *view* privilege on the object and the *link* privilege on all the links contained in the object.

Note that we have chosen to distinguish between *link* and *view* privileges because such difference makes it possible to grant users access to the information in an object (or in some of its slots) without disclosing the relationships of this object to other objects. For example, to facilitate blind reviewing of research papers, the reviewers should be given access to the papers without revealing possible links to publications by the same author(s) of the paper being reviewed.

Authoring privileges allow users to modify the content of an object or create a link. We distinguish among the following authoring privileges: *refer*, to include a link in an object or in a slot, *append*, to write information in an object (or in some of its slots) without deleting any pre-

existing information,⁴ `update`, to modify the content of an object (or of some of its slots) and to include links in the object. Thus, the `update` privilege subsumes both the `refer` and the `append` privilege. Moreover, we assume that if a user has the `update` privilege on an object, then he/she can also delete the object. Note that a user holding the `refer` or the `update` privilege on an object can specify links from the object to any other object. No authorisation is needed on an object to specify links with that object as destination of the link. Elements of P are therefore related in that some privileges are subsumed by others (for instance, `view` and `link` are subsumed by `view-all`). For this reason, also the set of privileges P is organised into a hierarchy, called *privilege hierarchy*, by means of a partial order $<_P$. We say that a privilege p_j is a *sub-privilege* of privilege p_i if and only if the relation $p_i <_P p_j$ holds. This means that privilege p_j is subsumed by privilege p_i .

Table 2 summarises the privileges supported by the Milan model and their interactions.

Type	Privilege	Meaning
Browsing	<code>view</code>	to see the information in an object (or in some of its slots). Links are not displayed
	<code>link</code>	to see the existence of links
	<code>view-all</code>	to see the information in an object and all the links it contains. It is equivalent to the <code>view</code> privilege on the object and the <code>link</code> privilege on all the object links
Authoring	<code>refer</code>	to include a link in an object (or in some of its slots)
	<code>append</code>	to write information in an object (or in some of its slots) without deleting any pre-existing information
	<code>update</code>	to modify (or delete) the content of an object. It subsumes both the <code>refer</code> and the <code>append</code> privilege.

Table 2: Authorisation privileges and their semantics

5 The Milan Authorisation Model

In this section we illustrate the basic components of the Milan authorisation model. We first illustrate how authorisations can be specified in the Milan model, then we discuss how to deal with authorisation conflicts.

5.1 Access Authorisations

In the Milan model both positive and negative authorisations can be specified. Positive authorisations indicate access permissions, whereas negative authorisations indicate access denials. Authorisations apply to concepts, objects, slots within objects, as well as to links. Access authorisations are formally defined as follows.

⁴ The `append` privilege can be used to authorise users to annotate a given object, without modifying its content.

Definition 5.1 (Authorisation) An authorisation is a 4-tuple $(\text{crd-spec}, \text{ent-spec}, \text{priv}, \text{sign})$ where crd-spec is a credential specification denoting the users to whom the authorisation is granted, ent-spec is an entity specification denoting the contents, objects, and/or the slots or links to which the authorisation refers, $\text{priv} \in P$ is the privilege for which the authorisation is granted, and $\text{sign} \in \{+, -\}$ indicates whether the authorisation is positive (+) or negative (-).

Tuple $(\text{crd-spec}, \text{ent-spec}, \text{priv}, +)$ states that the users denoted by crd-spec have been granted the privilege priv on the objects and/or slots or links denoted by ent-spec . Tuple $(\text{crd-spec}, \text{ent-spec}, \text{priv}, -)$ states that the users denoted by crd-spec have been forbidden privilege priv on the objects (slots or links) denoted by ent-spec . Since the view-all privilege allows one to see all information in an object, authorisations with $\text{priv} = \text{view-all}$ have necessarily ent-spec of the form $\{\text{co-spec} \mid \text{label-spec}\}.\text{slot-spec}$, with $\text{slot-spec} = \emptyset$ (cf. Definition 4.3). Moreover, authorisations with $\text{priv} = \text{view}$ have necessarily ent-spec of the form $\{\text{co-spec} \mid \text{label-spec}\}.\text{slot-spec}$, with slot-spec possibly empty, whereas authorisations with $\text{priv} = \text{link}$ have ent-spec of the form $\{\text{co-spec} \mid \text{label-spec}\}.\text{slot-spec}$ with $\text{slot-spec} = \emptyset$, or of the form link-spec . Finally, authorisations for an authoring privilege have ent-spec of the form $\{\text{co-spec} \mid \text{label-spec}\}.\text{slot-spec}$, with slot-spec possibly empty. All the above conditions are summarised by Table 3.

Privilege	Entity Specification
view-all	$\{\text{co-spec} \mid \text{label-spec}\}.\text{slot-spec}$ with $\text{slot-spec} = \emptyset$
view	$\{\text{co-spec} \mid \text{label-spec}\}.\text{slot-spec}$
link	$\{\text{co-spec} \mid \text{label-spec}\}.\text{slot-spec}$ with $\text{slot-spec} = \emptyset$ or link-spec
authoring	$\{\text{co-spec} \mid \text{label-spec}\}.\text{slot-spec}$

Table 3: Authorisation privileges and entity specifications

Example 5.1 The following are examples of authorisations:

- $A_1 = (\text{LLOC employee}(X), \text{World Law Bulletin.Blue page report}, \text{view-all}, -)$ prevents all the users with an associated credential of type LLOC employee from seeing the information in the *Blue page report* of the *World Law Bulletin*;
- $A_2 = ((\text{NML employee}(X) \vee \text{legal research directorate employee}(X)), (\text{Imports Tax} \vee \text{Tax Incentive}), \text{view}, +)$ authorises users with an associated credential of type NML employee or $\text{legal research directorate employee}$ to see the information in all the objects containing the concept *Imports Tax* or the concept *Tax Incentive*, but not the links contained in such objects;
- $A_3 = (\text{NML employee}(X), \text{Import Controls}, \text{link}, -)$ prevents NML employee s from seeing links in objects containing the concept *Import Controls*;
- $A_4 = ((\text{legal research analyst}(X) \wedge X.\text{national origin} = \text{Italy}), \text{Attorney's fees and Litigation Expenses in Selected Foreign Nations.Italian part}, \text{update}, +)$ authorises all the $\text{legal research analyst}$ s whose country of origin is *Italy* to modify the part pertaining to *Italy* of the report *Attorney's fees and Litigation Expenses in Selected Foreign Nations*.

- $A_5 = (\text{LLOC employee}(X), (\text{nudity} > 2), \{\text{Introduction, America}\}, \text{view-all}, -)$ prevents all the users with an associated credential of type `LLOC employee` from seeing the information contained in slots `Introduction` and `America` of the objects where the category `nudity` is labelled with a value greater than 2;
- $A_6 = ((\text{NML employee}(X) \vee \text{legal research directorate employee}(X)), (\text{sex} \geq 2), \text{view}, +)$ authorises all the users with an associated credential of type `NML employee` or `legal research directorate employee` to see the information contained in all the objects where the category `sex` is labelled with a value greater than or equal to 2;
- $A_7 = (\text{NML employee}(X), (\text{violence} = 3), \text{link}, -)$ authorises all users with an associated credential of type `NML employee` to see the information contained in all the objects where the category `violence` is labelled with a value equal to 3;
- $A_8 = ((\text{legal research analyst}(X) \wedge X.\text{national origin} = \text{Italy}), (\text{language} > 2), \text{update}, +)$ authorises all the users with an associated credential of type `legal research analyst` and whose country of origin is Italy to modify the information contained in all the objects where the category `language` is labelled with a value greater than 2;

In the following, the term *Authorisation Base* (\mathcal{AB}) is used to denote a set of authorisations. Moreover, given an authorisation $A = (\text{crd-spec}, \text{ent-spec}, \text{priv}, \text{sign})$ we denote with $\text{crd-spec}(A)$, $\text{ent-spec}(A)$, $\text{priv}(A)$, and $\text{sign}(A)$ the credential specification, the entity specification, the privilege, and the sign of A , respectively.

According to Definition 5.1, a single authorisation may be used to authorise a set of users to exercise the same privilege on the same objects, slots or links. These users can be explicitly specified, by listing their identifiers, or implicitly denoted by means of a credential expression. In order to enforce access control, it is necessary to identify, given an authorisation, the set of users to which it applies. A critical issue is represented by the possible presence of null values in the credentials associated with users. Indeed, suppose that the credential expression $\text{legal research analyst}(X) \wedge X.\text{age} > 18$, identifying all the users with an associated credential of type `legal research analyst` and whose age is greater than 18, appears in an authorisation A . Consider a user u , with an associated credential of type `legal research analyst`, such that no value is specified for attribute `age` in his/her credential. The problem arises in determining whether authorisation A applies to user u , since we have no information on his/her age. In the Milan model we take the most conservative approach: if A is positive, then A does not apply to user u ; by contrast if A is negative then it holds for user u also.

These concepts are formalised by the notion of *denoted users* which is the set of users to which an authorisation applies. This set is defined by using functions $\text{Denotes}()$ and $\text{Undef}()$, introduced in subsection 4.2.2.

Definition 5.2 (Denoted Users) Let A be an authorisation. The set of users to which A applies, denoted as U_A , is defined as follows:

- if $\text{crd-spec}(A) \in 2^U$, then $U_A = \text{crd-spec}(A)$;
- if $\text{crd-spec}(A) \in \mathcal{CE}$, then:
 - if $\text{sign}(A) = '+'$, then $U_A = \text{Denotes}(\text{crd-spec}(A))$;
 - if $\text{sign}(A) = '-'$, then $U_A = \text{Denotes}(\text{crd-spec}(A)) \cup \text{Undef}(\text{crd-spec}(A))$.

Example 5.2 Consider the authorisations of Example 5.1. Let $\mathcal{CB} = \{c_1, c_2\}$, where c_1, c_2 are the credentials of Example 4.7. $U_{A1} = \{\text{Ann}\}$, $U_{A2} = \{\text{Ann}\}$, $U_{A3} = U_{A4} = \emptyset$.

Similarly, we denote with *denoted objects*, the set of IDs of objects to which an authorisation applies. This set is formally defined as follows.

Definition 5.3 (Denoted Objects) Let A be an authorisation. The set of IDs of objects to which A applies, denoted as O_A , is defined as follows:

- if $\text{co-spec}(\text{ent-spec}(A)) \in 2^{OI}$, then $O_A = \text{co-spec}(\text{ent-spec}(A))$;
- if $\text{co-spec}(\text{ent-spec}(A))$ is a conceptual expression, then $O_A = \text{Obj}_c(\text{co-spec}(\text{ent-spec}(A)))$;
- if $\text{label-spec}(\text{ent-spec}(A))$ is a label condition, then $O_A = \text{Obj}_l(\text{label-spec}(\text{ent-spec}(A)))$;
- if $\{\text{co-spec}(\text{ent-spec}(A)) \mid \text{label-spec}(\text{ent-spec}(A))\} = \emptyset$, then:
 1. if $\text{slot-spec}(\text{ent-spec}(A)) \neq \emptyset$, then $O_A = \{i \mid (i, \text{slots}, \text{links}, \{\text{concepts} \mid \text{label}\}) \in \mathcal{OB}, i = i \text{ and } \text{slots} \cap \text{slot-spec}(\text{ent-spec}(A)) \neq \emptyset\}$;
 2. if $\text{link-spec}(\text{ent-spec}(A)) \neq \emptyset$, then $O_A = \{i \mid (i, \text{slots}, \text{links}, \{\text{concepts} \mid \text{label}\}) \in \mathcal{OB}, i = i \text{ and } \text{links} \cap \text{link-spec}(\text{ent-spec}(A)) \neq \emptyset\}$.

Example 5.3 Consider objects O_1 and O_2 of Example 4.1 and suppose an authorisation A_1 such that $\text{ent-spec}(A_1) = \{\text{Import-Export}\}$ and an authorisation A_2 such that $\text{ent-spec}(A_2) = \{\text{violence} \leq 2\}$. Since $C(O_1) = \{\text{Import Controls, Imports Tax, Taxation, Import-Export, GLIN Legal Document}\}$ and $Cat(\text{label-spec}(\text{ent-spec}(A_2))) = \{\text{violence}\}$, then $O_1 \in O_{A_1}$ and $O_2 \in O_{A_2}$.

5.2 Authorisation Propagation and Conflict Resolution

The fact that concepts, credential types, and privileges are hierarchically structured impacts the authorisations a user possesses, in that authorisations (both positive and negative) propagate along these hierarchies according to a set of pre-defined rules. As an example, suppose that a negative authorisation is given to all users with an associated credential of type `employee` and whose age is greater than 18. Then, it is reasonable to assume that this denial propagates to all the `legal research analyst` older than 18, since the credential type `legal research analyst` is a specialisation of `employee`. Therefore, in the Milan model, an authorisation given to all the users with a credential of a given type propagates to all the users whose credential is of a subtype of the credential-type which appears in the authorisation.

Similarly, authorisations propagate along the conceptual hierarchy, in that an authorisation given on the objects containing a given concept c implies an analogous authorisation on all objects containing a concept more specific than c . For instance, if a user has an authorisation to access all objects related to the concept `Taxation`, then he/she can also access all the objects related to the concept `Tax Exemption`.

Finally, an authorisation for a privilege p implies an analogous authorisation for all the privileges subsumed by p , that is, for all the privileges p' such that $p' <_p p$.

Example 5.4 Consider the authorisations of Example 5.1 and the credential-type hierarchy of Figure 3. Authorisation A_1 implies an analogous authorisation for all the users with a credential of type `legal research directorate employee, European division employee, eastern division employee and legal research analyst`.

Note that authorisation propagation is very useful since it is a means to concisely express a set of related authorisations. However, the Milan model allows one to specify exceptions with respect to

how authorisations propagate along the hierarchies by issuing proper negative authorisations. Although negative authorisations greatly augment the expressive power of the model, they have the drawback of introducing potential conflicts since a user may simultaneously hold both a negative and a positive authorisation for the same privilege on the same object, slot or link. However, we do not consider the simultaneous presence of conflicting authorisations as an inconsistency, rather we define a conflict resolution policy to deal with conflicting authorisations. Our conflict resolution policy keeps into account the conceptual, credential type, and privilege hierarchy, and is based on the concept of *most specific authorisation* (first introduced in [4]). The idea is that authorisations specified on lower level elements of the hierarchies prevail over authorisations specified on upper level elements. Moreover, since the Milan model supports more than one hierarchy, our conflict resolution policy establishes a priority order among the hierarchies, used when dealing with conflicting authorisations. More precisely, our conflict resolution policy is based on the following principles:

1. authorisations explicitly given to users (that is, authorisations in which the user identifiers explicitly appear) have the highest priority;
2. in all the other cases, the most specific authorisations, with respect to the credential type hierarchy, are considered as prevailing;
3. otherwise, the most specific authorisations, with respect to the conceptual hierarchy, are considered as prevailing;
4. when conflicts are not solved by the above rules, we consider the privilege hierarchy in that the authorisations specified for the most specific privileges are considered as prevailing.
5. finally, when conflicts are not solved by the principles above, negative authorisations take precedence.

Example 5.5 Consider the credential-type hierarchy in Figure 3, the conceptual hierarchy in Figure 1 and the authorisations of Example 5.1:

- A_2 prevails over $(\text{employee}(X), (\text{Imports Tax} \vee \text{Tax Incentive}), \text{view}, -)$ since $\text{NML employee}(X) <_{CT} \text{employee}$, and $\text{legal research directorate employee} <_{CT} \text{employee}$;
- A_1 prevails over $(\text{LLOC employee}(X), \text{World Law Bulletin.Blue page report}, \text{view-all}, +)$, in that the two authorisations differ only for the sign. In such case the most conservative approach is taken and thus we consider the negative authorisation as prevailing (see point 5. above);
- A_2 prevails over $(\text{NML employee}, \text{Import-Export} \vee \text{Taxation}, \text{view}, -)$, since $\text{Tax Incentive} <_{CP} \text{Taxation}$ and $\text{Imports Tax} <_{CP} \text{Import-Export}$.
- A_6 prevails over $(\text{employee}(X), (\text{sex} \geq 2), \text{view}, -)$ since $\text{NML employee}(X) <_{CT} \text{employee}$, and $\text{legal research directorate employee} <_{CT} \text{employee}$;
- A_5 prevails over $(\text{LLOC employee}(X), (\text{nudity} > 2), \{\text{Introduction}, \text{America}\}, \text{view-all}, +)$, in that the two authorisations differ only for the sign. In such case the most conservative approach is taken and thus we consider the negative authorisation as prevailing (see point 5. above);

Our conflict resolution policy is formalised by the concept of *stronger authorisations*. Consider two authorisations A_1 and A_2 and a privilege p such that $\text{priv}(A_i) = p$ or $p <_P \text{priv}(A_i)$, $i = 1, 2$. Suppose there exists a user u and an object dlo such that $u \in U_{A_1} \cap U_{A_2}$ and $i(dlo) \in O_{A_1} \cap O_{A_2}$. Intuitively, A_1 is stronger than A_2 wrt u and dlo , if whenever u requires to exercise privilege p on object dlo , authorisation A_1 prevails over authorisation A_2 , according to our conflict resolution policy.

Before formally defining the notion of stronger authorisation we need to introduce some preliminary definitions.

Definition 5.4 (Stronger Credential Specification) Let u be a user. Let A_1 and A_2 be two authorisations such that $u \in U_{A_1} \cap U_{A_2}$. $\text{crd-spec}(A_1)$ is stronger than $\text{crd-spec}(A_2)$ wrt user u , written $\text{crd-spec}(A_1) >_u \text{crd-spec}(A_2)$, iff one of the following conditions holds: *i*) $\text{crd-spec}(A_1) \in 2^U$ and $\text{crd-spec}(A_2) \in \mathcal{CE}$; *ii*) $\text{crd-spec}(A_1), \text{crd-spec}(A_2) \in \mathcal{CE}$ and $\forall ct_2 \in C_types(\text{crd-spec}(A_2)) \cap CT(u) \exists ct_1 \in C_types(\text{crd-spec}(A_1)) \cap CT(u)$ such that $ct_1 <_{CT} ct_2$.

Example 5.6 Let u be a user with a credential of type `LLOC employee` and with age greater than 18. Thus, according to the credential-type hierarchy in Figure 3, $CT(u) = \{\text{LLOC employee}, \text{employee}, \top_{CT}\}$. Hence:

- $\text{LLOC employee}(X) >_u (\text{employee}(X) \wedge X.\text{age} > 18)$;
- $\text{LLOC employee}(X) >_u X.\text{age} > 18$;
- $\{u_1, u\} >_u \text{LLOC employee}(X)$.

Definition 5.5 (Stronger Entity Specification) Let dlo be an object. Let A_1 and A_2 be two authorisations such that $i(dlo) \in O_{A_1} \cap O_{A_2}$. $\text{ent-spec}(A_1)$ is stronger than $\text{ent-spec}(A_2)$ wrt object dlo , written $\text{ent-spec}(A_1) >_{dlo} \text{ent-spec}(A_2)$, iff one of the following conditions holds:

1. $\text{co-spec}(\text{ent-spec}(A_1)) \in 2^{OI}$, $\text{co-spec}(\text{ent-spec}(A_2)) \in 2^{OI}$, $\text{ent-spec}(A_1)$ and $\text{ent-spec}(A_2)$ are of the form $\text{ent-spec.slot-spec}$ and $\text{slot-spec}(\text{ent-spec}(A_1)) \neq \emptyset$, whereas $\text{slot-spec}(\text{ent-spec}(A_2)) = \emptyset$;
2. $\text{co-spec}(\text{ent-spec}(A_1)) \in 2^{OI}$ and $\text{co-spec}(\text{ent-spec}(A_2))$ is a conceptual expression;
3. $\text{co-spec}(\text{ent-spec}(A_1))$ and $\text{co-spec}(\text{ent-spec}(A_2))$ are conceptual expressions and $\forall cp_2 \in \text{Concepts}(\text{co-spec}(\text{ent-spec}(A_2))) \cap C(dlo) \exists cp_1 \in \text{Concepts}(\text{co-spec}(\text{ent-spec}(A_1))) \cap C(dlo)$ such that $cp_1 <_{CP} cp_2$;
4. $\text{co-spec}(\text{ent-spec}(A_1))$ and $\text{co-spec}(\text{ent-spec}(A_2))$ are conceptual expressions such that the condition in point 3) does not hold, and $\text{slot-spec}(\text{ent-spec}(A_1)) \neq \emptyset$, whereas $\text{slot-spec}(\text{ent-spec}(A_2)) = \emptyset$;
5. $\text{ent-spec}(A_1)$ is of the form link-spec , whereas $\text{ent-spec}(A_2)$ is of the form co-spec.slot-spec , with $\text{slot-spec} = \emptyset$.

Example 5.7 Consider the conceptual hierarchy of Figure 1. Let dlo be an object, and let $es_1 = (\text{Tax Exemption} \vee \text{Import-Export})$ and $es_2 = (\text{Taxation} \vee \text{Import Controls})$ be two entity specifications.

- If $\text{concepts}(dlo) = \{\text{Tax Exemption}\}$, then $C(dlo) = \{\text{Tax Exemption}, \text{Taxation}, \text{GLIN Legal Document}\}$, $\text{Concepts}(es_1) = \{\text{Tax Exemption}, \text{Import-Export}\}$, $\text{Concepts}(es_2) = \{\text{Taxation}, \text{Import Controls}\}$. Thus, $es_1 >_{dlo} es_2$, since $\text{Tax Exemption} <_{CP} \text{Taxation}$;
- if $\text{concepts}(dlo) = \{\text{Taxation}, \text{Import Controls}\}$, then $C(dlo) = \{\text{Taxation}, \text{GLIN Legal Document}, \text{Import Controls}, \text{Import-Export}\}$, and thus $es_2 >_{dlo} es_1$;
- if $\text{concepts}(dlo) = \{\text{Tax Exemption}, \text{Import Controls}\}$, then $C(dlo) = \{\text{Tax Exemption}, \text{Taxation}, \text{GLIN Legal Document}, \text{Import Controls}, \text{Import-Export}\}$, and thus es_1 and es_2 are incomparable wrt $>_{dlo}$.

We are now ready to introduce the notion of stronger authorisation.

Definition 5.6 (Stronger Authorisation) Let u be a user, and let dlo be an object. Let A_1, A_2 be two authorisations such that $u \in U_{A_1} \cap U_{A_2}$ and $i(dlo) \in O_{A_1} \cap O_{A_2}$. Authorisation A_1 is stronger than authorisation A_2 wrt u and dlo , written $A_1 >_{u,dlo} A_2$, iff one of the following conditions holds:

1. $crd-spec(A_1) >_u crd-spec(A_2)$;
2. $crd-spec(A_1)$ and $crd-spec(A_2)$ are incomparable wrt the $>_u$ relation, and $ent-spec(A_1) >_{dlo} ent-spec(A_2)$;
3. $crd-spec(A_1)$ and $crd-spec(A_2)$ are incomparable wrt the $>_u$ relation, $ent-spec(A_1)$ and $ent-spec(A_2)$ are incomparable wrt the $>_{dlo}$ relation, and $priv(A_1) <_p priv(A_2)$;
4. $crd-spec(A_1)$ and $crd-spec(A_2)$ are incomparable wrt the $>_u$ relation, $ent-spec(A_1)$ and $ent-spec(A_2)$ are incomparable wrt the $>_{dlo}$ relation, $priv(A_1)$ and $priv(A_2)$ are incomparable wrt the $<_p$ relation, and $sign(A_1) = '-'$, whereas $sign(A_2) = '+'$.

We can finally introduce the notion of valid authorisations, that is, the strongest authorisations in \mathcal{AB} . We distinguish between authorisations valid for a slot or a link.

Definition 5.7 (Valid Slot Authorisation) Let u be user and let dlo be an object. Let s be a slot of dlo . Let \mathcal{AB} be an authorisation base and let A be an authorisation in \mathcal{AB} such that $priv(A) \neq link$ and either $slot-spec(ent-spec(A)) = \emptyset$ or $s \in slot-spec(ent-spec(A))$. Authorisation A is valid for user u wrt slot s of object dlo if and only if no authorisation $A' \in \mathcal{AB}$ exists such that $sign(A') \neq sign(A)$, $priv(A') \neq link$, $A' >_{u,dlo} A$, and either $slot-spec(ent-spec(A')) = \emptyset$ or $s \in slot-spec(ent-spec(A'))$.

Definition 5.8 (Valid Link Authorisation) Let u be a user and let dlo be an object. Let l be a link of dlo . Let \mathcal{AB} be an authorisation base and let A be an authorisation in \mathcal{AB} such that $priv(A) \neq view$ and either $ent-spec(A)$ is of the form $\{co-spec \mid label-spec\}.slot-spec$ with $slot-spec = \emptyset$, or $ent-spec(A)$ is of the form $link-spec$ with $l \in link-spec$. Authorisation A is valid for user u wrt link l of object dlo if and only if no authorisation $A' \in \mathcal{AB}$ exists such that $priv(A') \neq view$, $sign(A') \neq sign(A)$, $A' >_{u,dlo} A$, and either $ent-spec(A')$ is of the form $co-spec.slot-spec$, with $slot-spec = \emptyset$, or $ent-spec(A')$ is of the form $link-spec$, with $l \in link-spec$.

6 The Milan Model Access Control

Once a user submits an access request along with the appropriate information, that request must be evaluated against the authorisations stored in the server \mathcal{AB} . Suppose that a user u wishes to exercise privilege p on objects dlo . At the server site, his/her access request can be represented as a triple (u, dlo, p) , where u is the user requesting the access, $dlo \in \mathcal{OB}$ is the object to which u requires the access, and p is the privilege that user u wishes to exercise on dlo . In the Milan model, one can authorise a user to exercise a privilege on a whole object or only selected portions (i.e., slots) and/or particular links within the object. This means that an access request can be *partially* authorised.

Given an authorisation base \mathcal{AB} and an access request (u, dlo, p) , we first need to identify the subset of the authorisations in \mathcal{AB} that must be evaluated in order to decide whether the access request can be authorised. This subset is formally identified by means of the notion of *authorisation base projection*, defined as follows.

Definition 6.1 (Authorisation Base Projection) Let (u, dlo, p) be an access request and let \mathcal{AB} be an authorisation base. The projection of \mathcal{AB} wrt (u, dlo, p) , denoted $\prod_{(u, dlo, p)}(\mathcal{AB})$, is the subset of \mathcal{AB} such that: $\forall A \in \mathcal{AB}, A \in \prod_{(u, dlo, p)}$ iff $u \in U_A, i(dlo) \in O_A$, and $\text{priv}(A) = p$ or $p <_p \text{priv}(A)$.

Thus, upon an access request the user is given a *view* of the object that contains only those slots and links for which he/she has a valid positive authorisation. In the case of totally authorised requests, the view is equal to the whole object. The notion of object view is formally defined as follows.

Definition 6.2 (Object View) Let (u, dlo, p) be an access request. The view of user u on object dlo wrt privilege p (denoted as $V_u^p(dlo)$) is the union of a set $S \subseteq \text{slots}(dlo)$ of slots and a set $L \subseteq \text{links}(dlo)$ of links such that:

- $\forall s \in \text{slots}(dlo), s \in S$ iff there exists in $\prod_{(u, dlo, p)}$ a positive authorisation A valid for user u wrt slot s of object dlo ;
- $\forall l \in \text{links}(dlo), l \in L$ iff there exists in $\prod_{(u, dlo, p)}$ a positive authorisation A valid for user u wrt link l of object dlo .

An algorithm enforcing access control is reported in Figure 4. Algorithm 6.1 receives as input an access request (u, dlo, p) and an Authorisation Base; it returns REJECT, if the access request cannot be authorised, it returns the view of user u on object dlo wrt privilege p , otherwise. Suppose that user u requires to access object dlo under privilege p . The algorithm makes use of two arrays, namely *Curr_slots* and *Curr_links*. Suppose that s_1, \dots, s_n are the slots in dlo , whereas l_1, \dots, l_m are the links in dlo . *Curr_slots* and *Curr_links* are incrementally updated and, the end of the algorithm, the i -th element of *Curr_slots* (*Curr_links*) is equal to '+' iff user u is authorised to access slot s_i (link l_i) of object dlo under privilege p . First, the algorithm computes (step 1) the projection of the authorisation base wrt the access request. If the projection is empty, the access is denied. Otherwise, in step 3 the authorisations in $\prod_{(u, dlo, p)}(\mathcal{AB})$ with the strongest credential specifications are considered. From these authorisations the strongest ones are selected. Procedure *Find-strong-auth* receives as input the set of authorisations in $\prod_{(u, dlo, p)}(\mathcal{AB})$ with the strongest credential specifications (these authorisations are contained into set *Act_Auth*), the object on which the access request is performed and the arrays *Curr_slots* and *Curr_links* computed till that point in the computation. For each slot s_i (resp. link l_i) of object dlo , procedure *Find_strong_auth* determines if there exists in *Act_Auth* a positive valid authorisation for user u wrt slot s_i (resp. link l_i) of object dlo . If such authorisation exists, then *Curr_slots*[i] (resp. *Curr_links*[i]) is set equal to '+'. Then, the authorisation in *Act_Auth* are removed from $\prod_{(u, dlo, p)}(\mathcal{AB})$, and the process is iteratively repeated until all the authorisations in $\prod_{(u, dlo, p)}(\mathcal{AB})$ have been considered. At the end of the algorithm, $V_u^p(dlo)$ is constructed from *Curr_slots* and *Curr_links*.

Algorithm 6.1 Access Control Algorithm

INPUT: 1) An access request (u,dlo,p) 2) The authorisation base \mathcal{AB}
OUTPUT: 1) $V_u^p(dlo)$, if $V_u^p(dlo) \neq \emptyset$, 2) REJECT, otherwise
METHOD:

1. Compute $\prod_{(u,dlo,p)}(\mathcal{AB})$
 2. $Curr_slots$, $Curr_links$ and $V_u^p(dlo)$ are initialized to be empty
 3. **If** $\prod_{(u,dlo,p)}(\mathcal{AB}) \neq \emptyset$
 Repeat
 $Act_Auth := \{A \mid A \in \prod_{(u,dlo,p)}(\mathcal{AB}), \text{ and } \nexists A' \in \prod_{(u,dlo,p)}(\mathcal{AB}) \text{ such that } crd\text{-spec}(A') >_u crd\text{-spec}(A)\}$
 If $Act_Auth \neq \emptyset$: *Find-strong-auth*($Act_Auth, dlo, Curr_slots, Curr_links$)
 Remove Act_Auth from $\prod_{(u,dlo,p)}(\mathcal{AB})$
 Until $\prod_{(u,dlo,p)}(\mathcal{AB}) = \emptyset$
 For each i such that $Curr_slots[i] = '+'$: Add s_i to $V_u^p(dlo)$
 For each i such that $Curr_links[i] = '+'$: Add l_i to $V_u^p(dlo)$
 If $V_u^p(dlo) \neq \emptyset$: **return** $V_u^p(dlo)$
 else: **return** REJECT
 else: **return** REJECT
-

Figure 4: Access Control Algorithm

The correctness of Algorithm 6.1 is stated by the following theorem.

Theorem 6.1 Algorithm 6.1 terminates. Moreover, given an access request (u,dlo,p) it returns $V_u^p(dlo)$ iff $V_u^p(dlo) \neq \emptyset$; it returns REJECT, otherwise.⁵

Example 6.1 Suppose that $\mathcal{AB} = \{A_1, A_2, A_3, A_4\}$, where A_1, \dots, A_4 are the authorisations of Example 5.1. Let Tom be a NML Employee and let dlo_1 be an object containing the concepts `Import Controls` and `Imports Tax`. Suppose that Tom issues the following access request $(Tom, dlo_1, view\text{-}all)$. By authorisations A_2 and A_3 , Tom is allowed to see all the information in dlo_1 , except for the links that dlo_1 eventually contains. Now suppose that the access request $(Helen, World\ Law\ Bulletin, view\text{-}all)$ is submitted to the system, where user Helen is a LLOC employee. By authorisation A_1 , Helen is returned a view of the World Law Bulletin which does not contain the Blue page report.

⁵ See [1] for the proof.

7 The Architecture of the Milan Model

We have implemented a prototype system for the Milan model. The prototype system has been developed on top of the Oracle 8.03 DBMS, using Delphi 3 client/server, and implements all the functions of the Milan model.

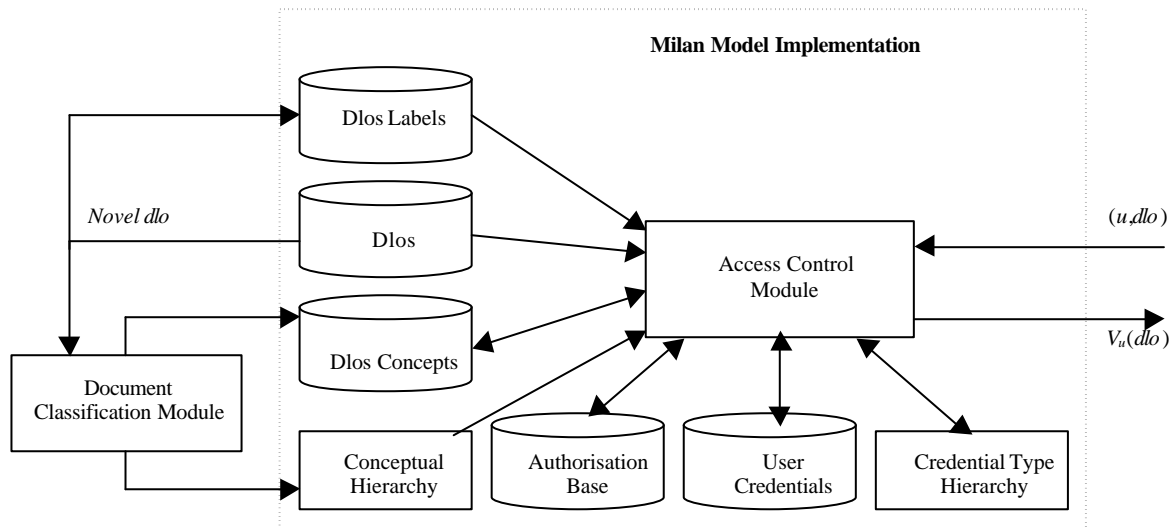


Figure 5: The Milan model architecture

The core of the system is the *Access Control Module*, which is in charge of performing access control to the DL objects with respect to the specified policies and the credentials associated with the user requesting the access. When a new document is acquired by the DL, if PICS labels are associated with it, they are stored into the system and used to perform access control. By contrast, if no labels are associated with the document, it is processed by the *Document Classification Module*, which a) extracts the concepts from the objects, b) builds a hierarchy of concepts and then c) classifies new documents into one or more concept-classes.

The Milan model allows the Security Officer (SO) to easily perform administrative operations such as the insertion and deletion of an authorisation, or to fill in the credential of a new user. For each administrative operation supported by the Milan model, the prototype provides a graphical interface that supports the SO in performing the administrative operation. For instance, new authorisations can be entered into the \mathcal{AB} using the screen shown in Figure 6. In this screen the SO enters general information on the new authorisation, such as the sign and the privilege of the authorisation, the type of its credential specification (that is, whether it is a set of user identifiers or a credential expression) and the type of the entity specification. After this preliminary information is entered, the SO must specify both the credential and the entity specification of the new authorisation. These operations are performed using a graphical interface that dynamically changes according to the type of credential and entity specification that must be entered.

Moreover, the prototype provides graphical tools to define and update the credential-type hierarchy and to structure *dlos* into slots. Similar functions are provided to insert/delete a link in a *dlo*.

The screenshot shows a 'New Authorization' dialog box with three tabs: 'Type and Sign', 'Subject', and 'Object'. The 'Type and Sign' tab is selected. The dialog is divided into several sections:

- Type of authorization:** Two radio buttons, 'Link authorization' and 'Object authorization'. 'Object authorization' is selected.
- Privilege:** Two radio buttons, 'View' and 'View all'. 'View' is selected.
- Sign:** A dropdown menu currently showing 'Positive'.
- Object specification:** Three radio buttons, 'Object identifiers', 'Conceptual expression', and 'Slot name only'. 'Object identifiers' is selected.
- Credential Specification:** Two radio buttons, 'User identifiers' and 'Credential expression'. 'User identifiers' is selected.

An 'Add Authorization' button is located at the bottom center of the dialog.

Figure 6: Authorisation specification in the Milan model prototype

8 Conclusions

In this deliverable we described the Milan model, a content-based access control mechanism for Digital Libraries (DL), supporting the PICS standard.

In the GLIN framework, the Milan model dealt with mere textual documents. All the same, the Milan model naturally extends to other media, since it is based on concepts associated with objects. In particular, our access control model can be coupled with any system able to associate information in form of concepts to different media objects, such as systems supporting metadata information [7]. Moreover, the Milan model can be intended as a *core mechanism* by which many protection policies can be enforced.

As we already said in the introduction of this deliverable, the Milan model will be used in the filtering component of the EUFORBIA prototype, in parallel with direct utilisation of NKRL EUFORBIA labels.

The concepts to be used as input to the Milan module will be extracted from the EUFORBIA labels represented in NKRL format. To do this, a set of generalised 'if-then' rules will analyse the fillers of some NKRL roles like OBJ(ect), TOPIC or CONTEXT according to the 'general meaning' expressed by the label (or set of labels) at hand. Note that the concepts utilised by the Milan model are, obviously, those pertaining to the H_CLASS hierarchy of NKRL.

In order to use the Milan model in the framework of the EUFORBIA project we need to

- rewrite the Milan module in Java;
- contribute to adapt H_CLASS to the EUFORBIA context, verifying in particular that the new concepts added are suitable for use in the Milan model;
- develop a graphical tool for specifying filtering rules.

References

- [1] N. Adam, V. Atluri, E. Bertino, E. Ferrari. A Content-based Authorization Model for Digital Libraries, to appear in IEEE Transactions on Knowledge and Data Engineering.
- [2] N. Adam and et al. Strategic Directions in Electronic Commerce and Digital Libraries: Towards a Digital Agora. *ACM Computing Surveys*, 28(4), 1996.
- [3] R. Holowczac. *Extractors for Digital Library Objects*. PhD thesis, Rutgers University, Department of MS/CIS, 1997.
- [4] T. Lunt. Access Control Policies: some Unanswered Questions. *Computer & Security*, 8(1), 1989.
- [5] *Platform for Internet Content Selection (PICS)*. <http://w3.org/PICS/>.
- [6] P. Resnick and J. Miller. *PICS: Internet access controls without censorship*. Communication of the ACM vol. 39 N° 10 pages 87-93, 1996.
- [7] V.S. Subrahmanian. *Principles of Multimedia Database Systems*. Morgan-Kaufann, 1997.
- [8] B. Thuraisingham. A Tutorial in Secure Database Systems. MITRE Technical Report, 1992.

Appendix

Table 4 lists, for the convenience of the reader, the symbols more frequently used in this deliverable and reports a brief explanation of their meaning.

Symbol	Meaning
OI, LI, CT, CI	a set of object, link, credential-type, credential identifiers
SN, CP, AN, LB	a set of slot, concept, attribute names and labels
$COEX, LC$	a set of conceptual expressions and label conditions
\mathcal{T}	a set of types
\mathcal{V}	the set of legal values for types in \mathcal{T}
$C(dlo), Cat(dlo)$	the set of concepts and label categories characterising object dlo
$\leq_{CP}, \leq_{CT}, \leq_P$	the conceptual, credential-type, and privilege hierarchy
OB, CB, AB	the object, credential base, authorisation base
$A(ct)$	the set of attributes of instances of the credential-type ct
$a(v)$	the value of attribute a
$CT(u)$	the set of credential types characterising user u
\mathcal{CE}	the set of credential expressions
$Denotes(ce)$	the set of user identifiers denoted by the credential expression ce
$Undef(ce)$	the identifiers of the users that do not satisfy ce because of null values in their credentials
$C_types(ce)$	the set of credential types in the credential expression ce
$Concepts(cx)$	the set of concepts appearing in the boolean expression of concepts cx
$Categories(lc)$	the set of categories appearing in the label condition lc
$Obj_c(cx)$	the identifiers of the objects whose associated concepts satisfy the conceptual expression cx
$Obj_l(lc)$	the identifiers of the objects whose associated labels satisfy the label condition lc
U_A	the set of users to which authorisation A applies
O_A	the IDs of the objects to which authorisation A applies
$crd-spec(A_1) >_u crd-spec(A_2)$	the credential specification of authorisation A_1 is stronger than the credential specification of authorisation A_2 wrt user u
$ent-spec(A_1) >_{dlo} ent-spec(A_2)$	the entity specification in authorisation A_1 is stronger than the entity specification in authorisation A_2 wrt object dlo
$A_1 >_{u,dlo} A_2$	authorisation A_1 is stronger than authorisation A_2 wrt user u and object dlo
$\Pi_{(u,dlo,p)}(\mathcal{AB})$	the projection of the authorisation base \mathcal{AB} wrt the access request (u,dlo,p)
$V_u^p(dlo)$	the view of user u on object dlo wrt privilege p

Table 4: Notation and terminology