Short Running Title: Access Control Policies for XML Document Sources
Contact Author: Dr. Elena Ferrari
Address: University of Milan
Dipartimento di Scienze dell'Informazione
Via Comelico 39/41
20135 Milano (Italy)
Phone: (+39) 02-55006230
Fax: (+39) 02-55006253
email: ferrarie@dsi.unimi.it
Specifying and Enforcing Access Control Policies for
XML Document Sources

Elisa Bertino¹, Silvana Castano¹, Elena Ferrarà¹, Marco Mesiti²

¹ Dipartimento di Scienze dell’Informazione
Università degli Studi di Milano
Via Comelico, 39/41 20135 Milano, Italy
{bertino,castano,ferrarie}@dsi.unimi.it

² Dipartimento di Informatica e Scienze dell’Informazione
Università degli Studi di Genova
Via Dodecaneso, 35 16146 Genova, Italy
mesiti@dsi.unige.it
Abstract

The Web is becoming the main information dissemination means in private and public organizations. As a consequence, several applications at both internet and intranet level need mechanisms to support a selective access to data available over the Web. In this context, developing an access control model, and related mechanism, in terms of XML (eXtensible Markup Language) is an important step, because XML is increasingly used as the language for representing information exchanged over the Web. In this paper, we propose access control policies and an associated model for XML documents, addressing peculiar protection requirements posed by XML. A first requirement is that varying protection granularity levels should be supported to guarantee a differentiated protection of document contents. A second requirement arises from the fact that XML documents do not always conform to a predefined document type. To cope with these requirements, the proposed model supports varying protection granularity levels, ranging from a set of documents, to a single document or specific document portion(s). Moreover, it allows the Security Administrator to choose different policies for documents not covered or only partially covered by the existing access control policies for document types. An access control mechanism for the enforcement of the proposed model is finally described.

Keywords: XML documents, XML protection requirements, XML access control policies, XML access control model, authorization propagation.
1 INTRODUCTION

The Web is becoming the main information dissemination means in private and public organizations, both at internal and external level. As a consequence, several applications at internet and intranet level need mechanisms supporting a selective access to data available over the Web. Thus, access control plays a crucial role in Web data management. In particular, access control must be tailored to the language used to describe the data and to the types of actions that can be executed on such data, such as navigation and browsing. In this respect, developing an access control model, and related mechanism, in terms of XML (eXtensible Markup Language) [W3C 1998a] is an important step, because XML is increasingly used as the language for representing information exchanged over the Web.

XML has recently emerged as the most relevant standardization effort in the area of document representation through markup languages. XML is a subset of SGML [ISO 1986]. Its goal is to provide a language for creating documents that is easier to learn and use than SGML and semantically richer than HTML [W3C 1998b]. The main benefits of XML with respect to HTML are related to the possibility of defining tags, nested document structures, and document types (called DTDs -Document Type Definitions-) describing the structure of a set of documents.

In this scenario, there is a strong need for models and mechanisms enabling the specification and enforcement of access control policies specifically tailored to XML [Bertino et al. 1999a]. Tools allowing the generation of HTML pages from XML documents are now available, which make even more important the specification of access restrictions for XML. The development of an access control model and mechanism for XML poses new requirements. A first requirement arises from the fact that XML documents can contain information at different degrees of sensitivity and thus varying protection granularity levels should be supported. In some cases, the same access control policy may apply to a set of documents. In other cases, different access control policies may apply to different portions of the same document. Many other intermediate situations may also arise. The access control mechanism must be flexible enough to support a spectrum of protection granularity levels. A second requirement arises from the fact that XML documents do not always conform to a predefined document type. Because it is most likely the case that access control policies are specified in terms of document types, situations where a document is not covered by the existing access control policies have to be managed properly. Since document exchange and acquisition processes may be frequent over the Web, the access control mechanism has to cope very well with such a situation.

In this paper, we propose an access control model addressing the above two requirements. The model supports varying protection granularity levels, ranging from a set of documents, to a specific document, to a document portion. The proposed model supports positive and negative authorizations as well as authorization propagation. In particular, different propagation options are introduced to support the Security
Administrator in specifying how an authorization defined on a document or document type should apply by
default to protection objects at a finer granularity level within the document or document type, respectively.
To model exceptions when necessary, the Security Administrator can define explicit positive/negative au-
thorizations overriding the “by default” principle associated with propagation. Furthermore, the proposed
model allows the Security Administrator to choose different policies to protect documents not covered or only
partially covered by the existing access control policies. To the best of our knowledge, the model proposed
in this paper is the first access control model for XML documents supporting both the above requirements.

The paper is organized as follows. Section 2 presents basic concepts of XML and a reference model we
use throughout the paper. Section 3 discusses protection requirements and access control policies for XML
documents. Section 4 presents an access control model for XML documents. Section 5 discusses problems
related to the protection of new documents entering an XML source. Section 6 deals with the access control
mechanism, while Section 7 discusses related work. Finally, Section 8 concludes the paper and outlines future
research directions.

2 BASIC CONCEPTS OF XML

The basic construct of an XML document is the element. Elements can be nested at any depth
and can contain other elements (subelements). An element contains a portion of the document delimited
by two tags: the start tag, at the beginning of the element, of the form <tag-name>, and the end tag, at
the end of the element, of the form </tag-name>. Empty elements of the form <tag-name/> are also
possible. An example of XML document containing information on a company department is shown in
Figure 1. This document provides information on name, address, resume, salary, and medical dossier of
employees of the production department. For each employee, the document records also information on
his/her manager. In the figure, the department element is an example of document element, that is, the
outermost element containing all the elements of the document. Employee, address, email are examples of
elements at different depth in the hierarchical structure of the XML document. The address element is an
example of element with subelements in that it contains street, tel, email elements. Street is an example
of element containing text (data content element), while email is an example of empty element.

A list of attributes can also be specified for an element. Attributes are of the form name = value, where
name is a label and value is a quoted string, and are listed within the start tag of the element. Attributes can
have different types allowing one to specify an element identifier (attributes of type ID often called id), links
to other elements of the document (attributes of type IDREF, referring to a single target, or IDREFS, referring
to multiple targets), or additional information about the element (e.g., attributes of type CDATA containing
textual information). For instance, with reference to Figure 1, the relationship existing between an employee
and his/her manager is modeled by an IDREF attribute named manager in the employee element, whose value is the id of the employee manager.

In the following, based on the recent works in the literature ([Deutsch et al. 1999; Milo and Zohar 1998]), we introduce a formal model of XML documents and DTD. Let $\mathcal{I}_E$ be a set of element identifiers, Label a set of element tags and attribute names, and Value a set of attribute/element values.

**Definition 1 (XML document).** An XML document is a tuple $d = (V_d, \bar{v}_d, E_d, \phi_{E_d})$, where:

- $V_d = V^e_d \cup V^a_d$ is a set of nodes representing elements and attributes, respectively. Each $v \in V^e_d$ has an associated element identifier $id_v \in \mathcal{I}_E$, whereas each $v \in V^a_d$ has an associated value $val \in Value$;

- $\bar{v}_d$ is a node representing the document element (called document root);

- $E_d = E^e_d \cup E^a_d \subseteq V_d \times V_d$ is a set of edges, where $e \in E^e_d$ is an edge representing an element-subelement relationship or a link between elements due to IDREF(s) attributes (called link edge), and $e \in E^a_d$ is an edge representing an element-attribute relationship;

- $\phi_{E_d} : E_d \rightarrow Label$ is the edge labeling function.

According to this model, an XML document is modeled as a labeled graph where nodes represent elements and attributes, and edges relationships between them. Figure 2 shows the graph representation of the XML
document reported in Figure 1. A node representing an element containing the element identifier (id), which can be the ID attribute value associated with the element, or can be automatically generated by the system, if no attribute of type ID is defined. A node representing an attribute is represented as a gray circle with its associated value. For simplicity, data content of an element is represented as a particular attribute of the element whose name is content and whose value is the element data content itself.

The graph contains edges representing the element-attribute and the element-subelement relationships, and link edges, representing links between elements introduced by attributes of types IDREF. Edges are labeled with the tag of the destination node (i.e., an element or an attribute) and are represented by solid lines, whereas link edges are labeled with the name of the corresponding IDREF attribute and are represented by dashed lines.

A document type declaration can be attached to XML documents, specifying the rules that XML documents may follow. These rules are collectively known as the Document Type Definition (DTD). As an example, Figure 3 shows the DTD for the document in Figure 2. A DTD is composed of two parts: the element declarations and the attribute list declarations. The element declarations part specifies the structure of the elements contained in the document. In particular, for an element it specifies its subelements (if any), their order, whether they are optional ('?'), whether they may occur more times ('*? or '+ with the usual meaning), and whether subelements are alternative with respect to each other ('|'), and/or the type of its data content. This type may be EMPTY if no content is allowed, ANY if all kind of content is allowed, or #PCDATA if only data content is allowed. The attribute list declarations part specifies, for each element, the list of its attributes, in terms of names, types, optionality clauses (#IMPLIED to denote an optional attribute, #REQUIRED to denote a mandatory one), and (possibly optional) default values.

1) An attribute of type IDREF is considered as a list of attributes of type IDREF.
\[\langle \text{DOCTYPE}\ \text{department}\rangle\
\langle \text{ELEMENT}\ \text{department}\ (\text{employee})^*\rangle\
\langle \text{ELEMENT}\ \text{employee}\ (\text{name}, \text{address}, \text{resume}, \text{salary}, \text{medical-dossier})\rangle\
\langle \text{ELEMENT}\ \text{name}\ (\text{name}, \text{surname})\rangle\
\langle \text{ELEMENT}\ \text{address}\ (\text{street}, \text{tel}*, \text{email})\rangle\
\langle \text{ELEMENT}\ \text{resume}\ (\text{education}, \text{previous-job}, \text{hobby}, \text{skills})\rangle\
\langle \text{ELEMENT}\ \text{salary}\ (\text{PCDATA})\rangle\
\langle \text{ELEMENT}\ \text{medical-dossier}\ \text{ANY}\rangle\
\langle \text{ELEMENT}\ \text{email}\ \text{EMPTY}\rangle\
\langle \text{ELEMENT}\ \text{hobby}\ (\text{PCDATA})\rangle\
\langle \text{ELEMENT}\ \text{skills}\ \text{ANY}\rangle\]
\[\langle \text{ATTLIST}\ \text{department}\ \text{id}\ \text{ID} \text{#REQUIRED}\rangle\]
\[\langle \text{ATTLIST}\ \text{employee}\ \text{id}\ \text{ID} \text{#REQUIRED}\ \text{manager}\ \text{IDREF} \text{#IMPLIED}\rangle\]
\[\langle \text{ATTLIST}\ \text{email}\ \text{mailto}\ \text{CDATA} \text{#IMPLIED}\rangle\]

Figure 3. An example of DTD.

Let \(\mathcal{I}_E\) be a set of DTD element identifiers, and \(\text{Label}^*\) be the set of strings obtained through the concatenation of names in \(\text{Label}\) and a symbol in \(*, +, \?\). Let \(\text{Type}\) be a set of types, \(\text{Type}=\{\text{EMPTY}, \text{ANY}, \text{PCDATA}, \text{ID}, \text{IDREF}, \text{IDREFS}, \text{CDATA}\}\). The notion of DTD is formally defined as follows.

**Definition 2** (DTD). A document type definition (DTD) is a tuple \(t=(V_t, \overline{v}_t, E_t, \phi_{E_t})\), where:

- \(V_t = V_t^e \cup V_t^a\) is a set of nodes representing elements and attribute types, respectively. Each \(v \in V_t^e\) has an associated DTD element identifier \(id_t \in \mathcal{I}_E\), whereas, each \(v \in V_t^a\) has an associated type \(t \in \text{Type}\);
- \(\overline{v}_t\) is the node representing the whole DTD element (called DTD root);
- \(E_t \subseteq V_t \times V_t\) is a set of edges, where \(e \in E_t\) represents the element-subelement or element-attribute relationship;
- \(\phi_{E_t} : E_t \rightarrow \text{Label}^* \cup \{\text{union}, \text{content}\}\) is the edge labeling function.

According to this model, a DTD is represented as a graph where elements are represented as white circles and attribute types as grey circles. Figure 4 shows the graph representation of the DTD of Figure 3. In a DTD graph, an edge label followed by \("*"\) or \("+"\) represents a repeatable element (e.g., \text{tel} in Figure 4). An edge label followed by \("?"\) represents an optional element (e.g., \text{hobby} in Figure 4). To represent a choice among alternative subelements of a given element in the DTD, an edge label \text{union} is introduced. A \text{union}-labeled edge connects the node representing the element and a (dummy) node representing the choice, connected to all the nodes representing alternative subelements. Since an element is represented as a node in the graph
representation, we use indifferently the terms *element* and *node* in the remainder of the paper. Moreover, we will refer to IDREF(S) attributes as link(s).

An XML source can contain several documents and DTDs. In particular, two kinds of documents can be found in an XML source, namely, *valid* and *well-formed* documents. A well-formed document is a document that follows the grammar rules of XML [W3C 1998a]. A valid document is a document conforming to a given DTD. Therefore, valid documents can be considered instances of a corresponding DTD in the source. For example, the document in Figure 2 is a valid document, since it conforms to the DTD in Figure 4.

3 PROTECTION REQUIREMENTS AND POLICIES FOR XML DOCUMENTS

In this section, we discuss protection requirements and access control policies for XML documents.

3.1 Protection requirements

Protection of XML documents in an XML source poses new requirements with respect to data protection in conventional databases. New protection requirements derive from both the graph-based structure of XML documents and the presence of DTDs, valid, and well-formed documents in a given source. In the following, we discuss these aspects in detail.

3.1.1 Fine-level authorization granularity

As we have seen in the previous section, an XML document is structured into elements and attributes. Elements can contain elements (subelements) in turn, resulting in a hierarchical structure. Moreover, XML documents are inter-linked since their elements can refer to other related elements in the document through IDREF(S) attributes. Very often, different (sub)elements, attribute(s), or link(s) within the same document
have varying protection requirements. For example, information on name subelement of the employee element in Figure 2 could be made available to everyone, while information on the medical-dossier subelement should be distributed only to selected subjects. As another example, information about employee in a department can be made available to everyone, whereas information about who is the manager of whom (represented as a link between employee elements) could be kept hidden from most subjects and made visible only to a restricted number of authorized subjects. To ensure a differentiated and appropriate protection of the contents of XML documents, a protection granularity different from the whole document is required.

As a consequence, authorizations for XML documents should be associated with protection objects at different granularity levels. In particular, the following protection object granularities are identified:

- document/DTD
- set of documents
- (sub)element
- attribute
- link

3.1.2 Propagation of authorizations

Protection of XML sources with a large number of documents and/or with documents having varying protection requirements, can lead to the definition of a huge number of authorizations. To limit the number of authorizations to be defined and maintained for documents in a source, the graph structure of XML documents can be exploited to enforce authorization propagation. The notion of propagation introduces an “apply by default” principle for authorizations, based on relationships holding between protection objects in the graph structure. The following relationships are exploitable for authorization propagation in the graph representing XML documents and/or DTDs:

- **Element-to-subelement**: according to this relationship, an authorization specified for an element propagates to its subelement(s). Since elements can be nested at any depth, it is desirable to have the possibility of controlling the depth at which authorizations should be propagated along the document/DTD graph (e.g., propagation from an element to all its directed subelements only; cascading propagation from an element to all its subelements at any depth in the graph; no propagation at all).

- **Element-to-attribute/link**: according to this relationship, an authorization specified for an element propagates to all its attributes and links, if defined.
A further relationship exploitable for authorization propagation is the one holding between a DTD and its associated document instances as follows:

- **DTD-to-instance**: according to this relationship, an authorization specified on a protection object at the DTD level propagates on the same protection object in all documents that are recognized as valid instances of the DTD.

The by default principle associated with propagation demands for the possibility of specifying exceptions on propagated authorizations. In analogy with security models for object-oriented DBMSs [Rabitti et al. 1991], exceptions can be managed by introducing both positive (i.e., permissions) and negative (i.e., denials) authorizations. We will discuss authorization propagation and exceptions for XML in Section 4.

### 3.1.3 Protection of valid and well-formed documents

The presence of DTDs and of valid and well-formed documents in a source suggests the specification of authorizations at two different levels of abstraction:

- **DTD (i.e., schema) level**: a DTD defines the structure of a set of XML documents and can be considered as a schema for them. In analogy with authorization models for conventional databases, authorizations (at different granularity levels) can thus be specified at the DTD level and propagated to all documents that are valid instances of the DTD, according to the DTD-to-instance relationship.

- **Document (i.e., instance) level**: authorizations (at different granularity levels) are defined at the document level and propagate on the considered document only, according to the element-to-subelement and the element-to-attribute/link relationships.

### 3.2 Access control policies for XML documents

Based on the protection requirements previously described, protection of XML documents can be enforced according to different discretionary policies, discussed in what follows.

#### 3.2.1 DTD-based policy

According to this policy, authorizations are associated with DTDs of an XML source and are propagated to DTD instances, because of the DTD-to-instance propagation relationship. Authorizations can be defined at different granularity levels on a DTD (i.e., document root, element, link, attribute level) to provide a different protection degree to the contents of valid documents instances of the DTD. By exploiting propagation, document protection can be enforced by specifying a variable number of authorizations on the DTD, depending on the protection requirements to be enforced. For example, subjects can be authorized to access
documents with homogeneous protection requirements by specifying only one authorization at the DTD level for the DTD associated with these documents, and by requiring that this authorization is propagated to all subelements, attribute, and links of the DTD, because of the element-to-subelement and element-to-attribute/link relationship, and also to all document instances of the DTD, because of the DTD-to-instance relationship. A DTD with heterogeneous protection requirements on its contained element(s) and/or attribute(s)/link(s), can be properly protected by specifying an authorization for the whole DTD without propagation, plus a number of additional authorizations for its subelement(s) and/or attribute(s)/link(s), to specify the specific access rights holding on them.

3.2.2 Document-based policy

According to this policy, authorizations are associated directly with an XML document. Each document is treated separately, by defining the most appropriate authorizations for its protection. Authorizations can be defined at different granularity levels on the document (i.e., document, element, link, attribute) to provide a different protection degree to the document contents. Authorizations defined on the document propagate only on the considered document according to the element-to-subelement and the element-to-attribute/link relationships.

For instance, if all information in a document has the same protection requirements, then only one authorization can be defined on the document element with a cascading propagation. By contrast, when different portions of the document have different protection requirements, an authorization at the document level can be defined with a minimal set of access rights and without propagation. In addition, a number of authorizations are defined on subelement(s), attribute(s), link(s) of the document, to specify the access rights holding on them.

3.2.3 Considerations

The DTD-based authorization policy is well suited for valid document protection, in that valid documents are always instances of some DTD in the source. This policy can be used also for well-formed document protection, provided that well-formed documents are first classified against available DTDs, with the goal of finding the “best matching” DTD, that is, the DTD capable of describing the structure of the well-formed document to be protected. If such a DTD is found, protection of a well-formed document can be enforced by propagating authorizations defined for the selected DTD to the document. We address issues related to the classification for well-formed document protection in more detail in Section 5.

The document-based authorization policy is useful when a conforming DTD is not found for a well-formed document, or when a classification process is not performed, or when a valid document has peculiar
protection requirements, in addition to the ones specified on its corresponding DTD.

Orthogonally, authorization propagation combined with the possibility of specifying both positive and negative authorizations is a means to limit the number of authorizations to be specified by allowing, at the same time, to enforce a strict control on particular crucial data. By using negative authorizations, it is possible for instance to limit the scope of the propagation of a positive authorization, and vice-versa. This can be useful, for instance, when access should be given to all the portions of a document, apart from few of them. In such a case, the document can be protected by specifying only two authorizations: a positive one at the document level with cascading propagation, and a negative one on only those portions to which the access should be denied. Such negative authorizations have the effect of blocking the propagation of the positive one.\(^2\)

4 AN ACCESS CONTROL MODEL FOR XML DOCUMENTS

In this section, we introduce a discretionary access control model for XML documents, enforcing the protection requirements and policies outlined in the previous section. The model is based on authorizations specifying the privileges that subjects can exercise on protection objects and their allowed propagation. In the model, we consider as subjects users holding accounts at the XML document source to be protected. Before introducing the proposed authorization format, we describe protection objects, privileges, and propagation options.

4.1 Authorization objects

To enforce the fine-level granularity requirement, authorization specification supports a wide set of protection objects, ranging from sets of XML documents to a specific portion of a document (such as a specific element or a specific attribute and/or link within an element). Thus, the specification of the objects to which an authorization applies requires three steps. First, it is necessary to identify a document/DTD (or a set of documents/DTDs) to which an authorization applies. Once the documents/DTDs to which an authorization applies have been specified, we need a notation to identify selected elements within the specified documents/DTDs. Finally, it is necessary to identify the portion of the selected elements to which the authorization applies (i.e., specific links and/or attributes).

Elements of an XML document (or a DTD) can be explicitly denoted by listing their identifiers, or can be implicitly specified by giving a path from the document (or DTD) root to the node representing the element. The use of wild cards (*) in the path allows one to identify more than one node. These possibilities are formalized by the notion of element specification, introduced next.

\(^2\)Conflicts between positive and negative authorizations are discussed in Section 4.
Definition 3 (Element specification). Let $d = (V, \delta, E, \phi_E)$ be an XML document or a DTD. An element specification for $d$, denoted by $\text{element-spec}$, is:

- a set of identifiers of nodes in $d$, that is, $\text{element-spec} = \{id_1, \ldots, id_n\}$, with $\{id_1, \ldots, id_n\} \subseteq \{id_v \mid v \in V\}$, or

- a path expression, that is, $\text{element-spec} = \text{path}_E \text{xpr}$, where $\text{path}_E \text{xpr}$ is specified according to the following grammar: $\text{path}_E \text{xpr} ::= * \mid \text{tag} \mid \text{path}_E \text{xpr}$. $\text{path}_E \text{xpr}$, where $\text{tag}$ is an edge label corresponding to a tag of an element in $V$. $\square$

For example, $\text{address.street}$, $\text{address.*}$, $\text{address.tel}$, and $\{\text{E123, E150}\}$ are examples of possible element specifications referring to the XML document in Figure 2 (the meaning of these element specifications will be explained in Example 1 below).

We are now ready to introduce the notion of protection object specification. A protection object specification denotes the object(s) to which an authorization applies. For the sake of clarity, we distinguish between specifications that refer to documents (denoted as document specifications) and specification that refer to DTDs (denoted as DTD specifications). In the following, recalling that $I_E$ and $I_{E1}$ denote the set of element identifiers and the set of DTD element identifiers respectively, we denote with $I_D \subseteq I_E$ the set of identifiers associated with document roots, and with $I_{D1} \subseteq I_{E1}$ the set of element identifiers associated with DTD roots in the source.

Definition 4 (Document specification). A document specification is of the form:

$$\text{doc-spec} [\text{element-spec}] [\text{set-of-attrs} \mid \text{set-of-links}]$$

where:

- $\text{doc-spec} \in I_D \cup \{*\}$ is an XML document identifier or the symbol $\*$. Symbol $\*$ is used to denote all documents in the source;

- $\text{element-spec}$ is an element specification for the document(s) denoted by $\text{doc-spec}$. If $\text{doc-spec} = \*$, then $\text{element-spec}$ can be an element specification for any document in the source. Element-spec is optional (denoted by square brackets in the notation).

- $\text{set-of-attrs}$ and $\text{set-of-links}$ identify a set of attributes and a set of links, respectively. Formally, if $\text{element-spec}$ is specified then let $\{v_1, \ldots, v_n\}$ be the set of elements whose identifiers are denoted by $\text{element-spec}$ and Attr$(v)$ and Link$(v)$ be the set of attributes and the set of links of a node $v \in \{v_1, \ldots, v_n\}$, respectively. Then, $\text{set-of-attrs} \subseteq \cup_{i=1}^n \text{Attr}(v_i)$, and $\text{set-of-links} \subseteq \cup_{i=1}^n \text{Link}(v_i)$. If $\text{element-spec}$ is omitted then $\text{set-of-attrs}$ and $\text{set-of-links}$ identify a set of attributes and a set of links.
of the document root(s) specified by doc-spec. Note that, set-of-links and set-of-attrs can be optional and they are not required to appear together in the specification (symbol |' in the notation).

The document specification doc-spec.element-spec denotes the elements identified by element-spec of the documents specified by doc-spec (including all their links and attributes). If doc-spec = *, the specification identifies all the elements denoted by element-spec, regardless of the documents in which they are contained. If element-spec is omitted, it identifies all the elements of the documents denoted by doc-spec.

**Example 1** Consider the document in Figure 2. The following are examples of document specifications:

- production: it specifies the whole document.

- production.{E123, E101}: it specifies employees identified by E123 and E101.

- production.employee.address.tel: it specifies the phone numbers of all the employees of the production department.

- production.employee.{manager}: it specifies the link between an employee and his/her manager.

**Definition 5** (DTD specification). A DTD specification is of the form:

\[ DTD-spec[element-spec] [set-of-attrs | set-of-links] \]

where:

- \( DTD-spec \in I_{D_1} \cup \{\#\} \) is a DTD identifier or the symbol #. Symbol # is used to denote all the DTDs in the source;

- element-spec is an element specification for the DTD denoted by DTD-spec. If DTD-spec = #, then elements-spec can be an element specification for any DTD in the source;

- set-of-attrs and set-of-links identifies a set of attributes and a set of links, respectively. Formally, if element-spec is specified then let \( \{v_1, \ldots, v_n\} \) be the set of elements whose identifiers are denoted by element-spec. Then, \( set-of-attrs \subseteq \bigcup_{i=1}^n Attr(v_i) \), and \( set-of-links \subseteq \bigcup_{i=1}^n Link(v_i) \). If element-spec is omitted then set-of-attrs and set-of-links identify a set of attributes and a set of links of the DTD roots denoted by dtd-spec.

In the following, we use the term protection object specification to refer to both document and DTD specifications. We use document specifications and DTD specifications when a distinction has to be made.
4.2 Authorization privileges

Our model supports two kinds of privileges: *browsing privileges* and *authoring privileges*. Browsing privileges allow subjects to read the information in an element or to navigate through its links. Two different browsing privileges are supported: *read*, and *navigate*. The *read* privilege authorizes a subject to view an element and/or (some of) its components. The *navigate* privilege authorizes a subject to see the existence of a specific link or of all the links in a given element and to navigate through them. Clearly, a subject can exercise the *navigate* privilege on a link only if he/she has the privilege to access the destination element of the link. Finally, the *browse all* privilege subsumes both the *navigate* and the *read* privilege, thus allowing a compact specification of access control policies. Note that we have chosen to distinguish between *read* and *navigate* privileges because in this way it is possible to grant subjects the access to an element without disclosing the links of this element with other elements. For example, a subject authorized for the *read* privilege on the document in Figure 2 can access the information on all the employees in the production department, but he/she cannot see the relationship between an employee and his/her manager. To acquire this information, the subject needs a *navigate* privilege on the *manager* attribute.

Authoring privileges allow subjects to modify (or delete) the content of an element or to append new information in an element. We support two distinct authoring privileges: *append* and *write*. The *append* privilege allows a subject to write information in an element (or in some of its parts) or to include a link in an element, without deleting any pre-existing information. By contrast, the *write* privilege allows a subject to modify the content of an element and to include links in the element. Thus, the *write* privilege subsumes the *append* privilege. Moreover, we assume that if a subject has the *write* privilege on an element, then he/she can also delete the element.

4.3 Authorization propagation options

To enforce the element-to-subelement propagation, we introduce different *propagation options* in the specification of authorizations. A propagation option specifies how authorizations on an element at a given level of the DTD/document hierarchy propagate to its lower level element(s). Let $e$ be an element specified within an authorization, we introduce the following propagation options:

- *cascade*: the authorization propagates to all the direct and indirect subelements of $e$;
- *first lev*: the authorization propagates to all the direct subelements of $e$;
- *no prop*: no authorization propagation is performed.

The element-to-attribute/link and DTD-to-instance authorization propagation are intrinsically guaranteed by the semantics of access privileges. Privileges we introduced are *polymorphic*, in that the same
privilege has a different semantics depending on the considered protection object specification and on the propagation option specified in the authorization. Consider, for instance, a read authorization with the cascade propagation option and suppose that the authorization refers to a document, that is, the protection object specification is a document specification. The following cases can arise:

- if the document specification denotes a set of attributes (or a set of links), then the authorization gives the privilege to read the attributes (or to view the links without activating them);

- if the document specification denotes an element (or a set of elements), then the read authorization gives the privilege to see all the information in the element (that is, attributes and links) and, additionally, it recursively applies to all the subelements of the specified element;

- finally, if the document specification denotes a document (or a set of documents), then the read authorization applies to all the elements contained in the document as well as to the document attributes and links.

By contrast, if the authorization is specified at the DTD level, that is, its protection object specification is a DTD specification, then we have the following cases:

- if the DTD specification denotes a set of attributes (or a set of links), then the authorization gives the privilege to read the attributes (or to view the links without activating them) in all the DTD instances;

- if the DTD specification denotes an element (or a set of elements) in a DTD, then the read authorization has the same effect as a read authorization on that element (or on the set of elements) for each instance of the DTD;

- if the DTD specification denotes a DTD, then it has the same effect as a read authorization given on all the DTD instances.

4.4 Authorization specification

Let \( S \) be the set of subjects, \( O \) the set of protection object specifications for documents and DTDs in a given XML source, and \( P \) the set of browsing and authoring privileges. We are now ready to introduce the definition of authorization.

**Definition 6 (Authorization).** An authorization for an XML document is a tuple

\[
< \text{subject}, \text{object-spec}, \text{priv}, \text{prop-opt}, \text{sign} >
\]

where: \( \text{subject} \in S \) is the subject to whom the authorization is granted; \( \text{object-spec} \in O \) is a protection object specification; \( \text{priv} \in P \) is the privilege for which the authorization is granted; \( \text{prop-opt} \in \{\text{cascade}, \)
FIRST LEVEL, NO PROP} is the propagation option associated with the authorization, and sign ∈ \{+, −\} is the sign of the authorization, stating if it is a permission (sign=+) or a prohibition (sign=−).

The following are examples of authorizations that can be specified in our model.

**Example 2** Consider the XML document in Figure 2. Authorization \(A_1 = (\text{Bob, production, read, cascade, +})\) allows Bob to read all the elements, attributes, and links contained in the document identified by production, but it does not authorize to navigate through the document links. Authorization \(A_2 = (\text{Tom, production, \{E123\}, navigate, no_prop, -})\) forbids Tom to see who is the manager of employee E123. Authorization \(A_3 = (\text{Meg, production, \{E123\}, write, first_lev, +})\) allows Meg to write (and/or append) information in the element identified by E123 and in all its directed subelements. Finally, authorization \(A_4 = (\text{Mary, \#.employee.medical-dossier, read, cascade, -})\), defined at DTD level, forbids Mary to read the medical-dossier element of any employee document in the source.

The possibility of specifying both positive and negative authorizations introduces potential conflicts among authorizations, in that a subject may have two authorizations for the same privilege on the same protection object but with different signs. These conflicting authorizations can be either explicit or derived through propagation. We do not consider the simultaneous presence of conflicting authorizations as an inconsistency; rather we define a conflict resolution policy which is based on the notion of most specific authorization. The conflict resolution policy enforced by our model is based on the following principles:

- authorizations specified at the document level prevail over authorizations specified at the DTD level;
- authorizations specified at a given level in the DTD/document hierarchy prevail over authorization specified at higher levels;
- when conflicts are not solved by the above rules, we consider as prevailing negative authorizations.

**Example 3** Consider the authorizations of Example 2, and suppose that authorization \(A_5 = (\text{Mary, production, \{E123\}, read, cascade, +})\) is additionally specified. Since according to our conflict resolution policy authorizations specified at the document level prevail over authorizations specified at the DTD level, \(A_5\) prevails over authorization \(A_4\) when applied to the element identified by E123. As a result, Mary can read all the information in element E123 and in all its subelements, including the medical-dossier subelement.

By contrast, suppose that authorization \(A_6 = (\text{Bob, production.employee.salary, read, no_prop, -})\) is specified. With respect to the salary element, \(A_6\) overwrites authorization \(A_1\), since it is directly specified on the salary element and not inherited from higher levels in the document hierarchy. As a result, Bob can read all the information in the production document, apart from those in the salary elements.
5 CLASSIFICATION OF XML DOCUMENTS FOR PROTECTION

An important issue for the application of access control policies to XML documents in a given source is related to the management of new, well-formed documents entering the source. Because it is most likely the case that authorization policies are specified in terms of DTDs, it is important to discover whether a well-formed document entering a source conforms to an existing DTD. In such a case, the document can be totally or partially covered by the policies defined for this DTD, and a DTD-based policy can be adopted. Otherwise, a document-based policy can be adopted to specify all required authorizations directly on the well-formed document. In order to apply DTD-based access control policy to a well-formed document entering a source, we propose to perform a classification process to identify the most appropriate DTD and, consequently, apply its associated access control policy.

Given an XML document \( \tilde{d} \) entering a source, it is submitted to a classification process to find, among the available DTDs in the source, the one that best conforms to \( \tilde{d} \). This requires the analysis of the graph structure of \( \tilde{d} \) and its comparison with the graph structure of existing DTDs in the source. Different algorithms have been proposed in the literature for the analysis of graph-based document models, for their classification against a type hierarchy [Bertino et al. 1999b], and for the extraction of their representative structure [Nestorov et al. 1998]. The access control policy that can be applied to \( \tilde{d} \) depends on the result of the classification process. Conformance, partial conformance, and no conformance cases are distinguished, as shown in Figure 5.

5.1 Conformance

As a result of the classification process, a DTD \( \tilde{D} \) is found that conforms \( \tilde{d} \), that is, for each element/attribute of \( \tilde{d} \) one corresponding matching element/attribute is found in \( \tilde{D} \). In this case, we say that \( \tilde{d} \) conforms to \( \tilde{D} \) and can be automatically classified as a valid instance of \( \tilde{D} \). Consequently, authorizations
specified for $\tilde{d}$ propagate also to $d$, according to the DTD-to-instance propagation.

5.2 Partial conformance

As a result of the classification process, a DTD $\tilde{D}$ is found that partially conforms $\tilde{d}$, that is, only for some elements/attributes of $\tilde{d}$ a corresponding matching element/attribute is found in $\tilde{D}$. In this case, authorizations defined for $\tilde{D}$ propagate to $\tilde{d}$ as in the previous case. Moreover, policies are required to establish authorizations for each element/attribute $c$ of $\tilde{d}$ for which no corresponding matching element/attribute is found in $\tilde{D}$. The following access control policies are envisaged:

1. Propagation-based policy: this policy states that authorizations for $c$ are derived by propagation from authorizations defined for an element $c'$ of $\tilde{D}$ such that $c$ is a subelement/attribute of $c'$. For example, consider a document $\tilde{d}$ whose resume element contains also the foreignLanguages subelement. $\tilde{d}$ has a partial conformance with respect to the DTD in Figure 3. If a subject can read the resume element of $\tilde{d}$, then, according to the propagation-based authorization policy, the read privilege can be propagated also to the foreignLanguages subelement of resume.

2. Affinity-based policy: this policy states that authorizations for $c$ are derived by propagation from authorizations defined on an element/attribute $c'$ of $\tilde{D}$ such that $c'$ has affinity with $c$. The affinity criterion captures the fact that the labels of $c$ and $c'$ denote semantically related information, based on semantic content of the labels using a domain ontology [Castano and De Antonellis 1999]. For example, suppose that a medical-record element is contained in the resume element of $\tilde{d}$. Then, $\tilde{d}$ has only a partial conformance with the DTD in Figure 3. According to the affinity-based policy, authorizations defined for the medical-dossier element in the DTD are propagated to the medical-record element, since the term medical-record is synonym of the term medical-dossier in the Wordnet ontology [Miller 1995].

3. Document-based policy: this policy states that authorizations for $c$ are defined from scratch. This policy is always applicable, and is mandatory when no other policy can be used for $c$. For example, suppose that $\tilde{d}$ has a partial conformance with the DTD presented in Figure 3 because the employee element contains also the evaluation subelement. For evaluation, neither the propagation-based policy nor the affinity-based policy can be applied. Consequently, explicit authorizations have to be defined for evaluation by the Security Administrator.

The above three policies offer a different degree of automation and support to the Security Administrator in case of partial conformance. The first two policies exploit the element/subelement and element/attribute relationships as well as the affinity notion to limit the manual activity required to the Security
Administrator to define authorizations for $c$. In both policies, the Security Administrator can interactively validate propagated authorizations for $c$ to check their suitability, and possibly override them with new authorizations if necessary. The third policy requires a manual definition of the authorizations for $c$. Policies 1 and 2 need suitable mechanisms (e.g., rule-based mechanisms [Bertino et al. 1996]) to enforce propagation, that is, to automatically derive all required authorizations for all the involved subjects.

5.3 No conformance

As a result of the classification process, no matching DTD for $d$ is found in the source. In this case, the document-based policy is adopted for $d$. The Security Administrator defines all authorizations necessary to enforce protection of $d$.

6 ACCESS CONTROL MECHANISM

In this section, we describe the access control mechanism supporting the model presented in Section 4. We first give an overview of the proposed mechanism, then we illustrate access control strategies and, finally, we deal with implementation issues.

6.1 Overview

Subjects can request access to XML documents under two different modalities: browsing and authoring. A subject requests a browsing access when he/she wants to access the document (and navigate its links), without modifying it, whereas, he/she requests an authoring access when he/she wants to modify the document. Access can be requested to (a specific portions of) a document or to all the instances (or portions of them) of a given DTD. Thus, an access request $r$ is represented as a triple $r = (s, obj-spec, acc-modality)$, where $s \in S$ is the subject requesting the access, $obj-spec \in O$ is the protection object specification denoting the documents (or portions of them) to which the access is requested, and $acc-modality \in \{browsing, authoring\}$ specifies whether the subject is requesting a browsing or an authoring access. We assume that the object specification in an access request always contains either a document or a DTD identifier. If the object specification contains a DTD identifier, the request applies to all the DTD instances, whereas if it contains a document identifier, it applies to a specific document. The goal of the access control mechanism is to verify whether a subject requesting access to one (or more) XML document(s) is authorized for the requested privilege, according to the authorizations in the authorization base (see Figure 6). When the access request $r$ is submitted, the access control mechanism checks which authorizations (both positive and negative) $s$ has on the requested document(s). Such authorizations can be either explicitly defined or implicitly derived using the propagation relationships (i.e., DTD-to-instance, element-to-subelement,
element-to-attribute/link). Based on authorizations held by $s$ its request can be totally authorized, partially authorized, or denied. In general $s$ has a view of requested protection object(s) that contains only those portions of the object(s) for which he/she has a positive applicable authorization.\footnote{By applicable we mean a positive authorization which is not overwritten by a negative one.} If the subject has more than one applicable positive authorization on the same protection object (for instance, the subject has a read authorization on an element and a navigate authorization on one of the element links), the one with the more powerful privilege is considered (in the above example, the navigate authorization is considered and the subject can view the link and traverse it).

In the case of totally authorized requests, the view coincides with the whole requested protection object(s). In case of partially authorized requests, only a view of the whole requested protection object is returned. When, no positive authorizations are found or all of them are overwritten by negative authorizations, the access is denied.

**Example 4** Given the authorizations of Examples 2 and 3, suppose that Bob requests a browsing access to the employee element identified by E123. The access control mechanism verifies which browsing authorizations Bob possesses on such element. Bob has a positive authorization (i.e., $A_1$) on the production element and a negative authorization (i.e., $A_6$) specified on the salary subelement. Thus, Bob is authorized to access all the information in E123, apart from those in the salary element. Now suppose that Mary requires a browsing access to all the instances of employee. Mary has a negative authorization (i.e., $A_4$), at DTD level for medical-dossier, that propagates to all the document instances in the source, and a positive authorization (i.e., $A_7$) which is directly specified on E123. As a result, Mary is granted the access to E123 only, including the salary subelement.

\[\triangle\]

6.2 Strategies for access request evaluation

Because of the graph-based structure of both documents and DTDs, an access request can be evaluated according to two different strategies: top-down and bottom-up. These strategies can be applied either at the
1. Let \( r = \{ s, \text{obj-spec, browsing} \} \) be an access request

2. If \text{obj-spec} is a DTD specification, then, let \( \text{ddl} \) be the specified DTD
   - Let \( D \) be the set of \( \text{ddl} \) instances for which a negative authorization has been specified
   - For each \( d \in D \): apply the bottom-up strategy on \( d \)
   - If a negative authorization has been specified for \( \text{ddl} \): apply the bottom-up strategy on \( \text{ddl} \)
   - Otherwise: \( \text{Select-strategy} (\text{ddl}) \)
   - Let \( D' \) be the set of instances of \( \text{ddl} \) not belonging to \( D \) that cannot be totally returned to \( s \)
   - For each \( d' \in D' \): \( \text{Select-strategy} (d') \)

3. If \text{obj-spec} is a document specification, then, let \( d \) be the specified document
   - If \( d \) is valid or \( d \) conforms to a DTD, then let \( \text{ddl} \) be the DTD of \( d \)
     - If a negative authorization has been specified for \( d \): apply the bottom-up strategy on \( d \)
     - If a negative authorization has been specified for \( \text{ddl} \) or \( \text{CI}(\text{ddl}) = \text{HET} \)
       apply the bottom-up strategy on \( \text{ddl} \)
     - Otherwise if all the authorizations specified for \( \text{ddl} \) are positive: \( \text{Select-strategy}(\text{ddl}) \)
     - If all the authorizations specified for \( d \) are positive: \( \text{Select-strategy}(d) \)
   - Otherwise:
     - If a negative authorization has been specified for \( d \): apply the bottom-up strategy on \( d \)
     - Otherwise: \( \text{Select-strategy}(d) \)

Figure 7. Access control strategy for browsing access requests.

DTD or at the document level. At the DTD level they have the following effects:

- **Top-down strategy**: the system checks the authorizations specified at the DTD level for the requested protection object(s), starting from the authorizations specified at the DTD root granularity level. If all appropriate permissions are found at this level, and no other explicit authorizations are defined at a finer granularity level, then the access is totally authorized. Otherwise, the system considers authorizations specified at the element granularity level, starting from the first level DTD subelements and going down (if necessary) through the hierarchical structure of the DTD, until an appropriate authorization is found or the attribute/link level is reached.

- **Bottom-up strategy**: this strategy evaluates the authorizations specified at the DTD level for the requested protection object(s), starting from the ones specified at the most specific granularity level of the DTD and going up through the DTD hierarchical structure until an appropriate authorization is found or the DTD root granularity level is reached.

At the document level, the top-down and bottom-up strategies have the same effect but referred to a document graph, instead of a DTD graph.

6.3 Considerations and implementation issues

In the worst case, both top-down and bottom-up strategies require traversing the whole DTD and/or document graph, which can be rather expensive. An important question is thus to decide when the top-down
Procedure `Select-strategy(target)`

1. If $C_l(target) = \text{HOM}$: apply the top-down strategy on target
2. If $C_l(target) = \text{HET}$: apply the bottom-up strategy on target
3. If $C_l(target) = \text{MIX}$: apply either the top-down or the bottom-up strategy on target

Figure 8. Procedure `Select-strategy`

strategy is preferable over the bottom-up strategy in terms of efficiency, or vice-versa. The answer is driven by two major factors: the presence of negative authorizations and the homogeneity of documents with respect to protection requirements. Bottom-up strategies are chosen when negative authorizations are specified at the DTD and/or document level. This is due to the policy we adopt to solve conflicts between negative and positive authorizations, which considers as prevailing the most specific authorization with respect to the DTD and document hierarchies.

In absence of negative authorizations, top-down strategies are preferable for documents which are homogeneous with respect to protection requirements, because authorizations for such kind of documents are usually specified once, at the DTD/document granularity level and then propagated. By contrast, bottom-up strategies are more efficient when documents are heterogeneous with respect to their protection needs, since a number of explicit authorizations can be specified at several levels in the graph to enforce such protection needs.

In order to support the selection of the most efficient strategy, both documents and DTDs in the source are classified at three levels: homogeneous (HOM), heterogeneous (HET), and mixed (MIX). The classification level is used to select the most appropriate access control strategy. The classification level assigned to a DTD is by default propagated to all its instances, unless the Security Administrator specifies differently (i.e., the Security Administrator can state that specific documents have a HET classification although their associated DTD is classified as HOM).

Figure 7 presents the access control strategy for a browsing access request (in the case of authoring access requests we proceed analogously). In Figure 7 we make use of procedure `Select-strategy`, illustrated in Figure 8, that given a DTD/document identifier (`target`) selects the best access control strategy, on the basis of the DTD/document classification level. In the selection of the best strategy we make use of function $C_l()$ which returns the classification level of a DTD/document.

## 7 RELATED WORK

The work presented in this paper has relationships with access control models and mechanisms developed for object-oriented DBMSs [Fernandez et al. 1994; Rabitti et al. 1991] and WWW documents [Samarati et al. 1996].
The models proposed in [Fernandez et al. 1994; Rabitti et al. 1991] are specifically tailored to an object-oriented DBMS storing conventional, structured data. As such, great attention has been devoted to concepts such as versions and composite objects, which are typical of an object-oriented context. Those models support concepts such as positive and negative authorizations, and authorization propagation. Our model also supports such concepts, even though it has a larger variety of authorization propagation options. In our model three different options are supported by which the Security Administrator can specify i) that an authorization defined at a given level in the hierarchy propagates to all lower levels, ii) that the propagation stops at the first level down in the hierarchy, or iii) that no propagation has to be enforced. By contrast, the model proposed in [Rabitti et al. 1991] has only one propagation policy, which is equivalent to our cascade option.

The main difference between our context and an object-oriented DBMS is, however, that in an object-oriented DBMS, each object totally conforms to a class. Consequently, access control policies specified at the class level automatically apply to all the class instances. By contrast, XML documents do not always conform to a predefined document type and thus the above mentioned automatic propagation cannot be always exploited. Thus, access control policies for documents that do not conform (or only partially conform) to a document type must be devised. This is the main innovative feature of our access control model which is not found in any access control model previously proposed for object-oriented DBMSs.

An access control model for WWW documents has been proposed in [Samarati et al. 1996]. In such a model, HTML documents are considered, organized as unstructured pages connected by links. Authorizations can be given either on the whole document or on selected portions within the document. Although we borrow from [Samarati et al. 1996] the idea of selectively granting access to a document (by authorizing a subject to see only some portions and/or links in the document), our work substantially differs from this proposal. Differences are due to the richer structure of XML documents with respect to HTML documents and to the possibility of attaching a DTD to an XML document, describing its structure. Such aspects require the definition and enforcement of more sophisticated access control policies, than the ones devised for HTML documents. The access control model proposed in [Samarati et al. 1996] has great limitations deriving from the fact that it is not based on a language able to semantically structuring the data, as in our model for XML. As such, administering authorizations is very difficult. In particular, if one wants to give access to portions of a document, he/she has to manually split the page into different slots on which different authorizations are given. This problem is completely overcome in our model because XML provides semantic information for various document components. Authorizations can thus be based on this semantic information.

Finally, an access control model for XML documents has been recently proposed [Damiani et al. 2000]. Such model is very similar to previous models for object-oriented databases and does not actually take into
account some peculiarities of XML. In particular, this model has two main shortcomings. The first one is that it does not consider the case of documents not conforming/partially conforming to a DTD. Therefore, the model does not provide any support to the Security Administrator for dealing with such documents. We would like to stress that such a support is crucial when dealing with XML documents over the Web. By contrast, our model provides a rich variety of options one can choose to deal with such documents. Second, the model proposed in [Damiani et al. 2000] does not provide access control modes specific to XML documents. It only provides the read access mode. By contrast, we provide a number of specialized access modes for browsing and authoring, which allow the Security Administrator to authorize a user to read the information in an element and/or to navigate through its links, or to modify/delete the content of an element, or to add/modify or delete one of the element links. As a final remark, the model proposed in [Damiani et al. 2000] is at a early stage of design and has not been implemented yet. By contrast, our model has been implemented and a prototype system called Author-\textit{X} is available [Bertino et al. 2000].

8 CONCLUDING REMARKS

In this paper, we have presented an access control model and an associated mechanism for XML document protection. Novel features of the model are related to the capability of dealing with the interlinked, hierarchical structure of XML documents and with documents only partially conforming to existing DTDs and associated access control policies. Positive and negative authorizations at different granularity levels are supported, to protect set of documents, a whole document as well as selected portion(s) of a document. Different options are introduced for authorization propagation, to flexibly control the depth at which authorizations defined at a given level of granularity in the document/DTD hierarchy propagate to lower levels. Authorization propagation combined with positive and negative authorizations allows the Security Administrator to limit the number of authorizations to be defined as well as to specify exceptions, by defining explicit authorizations overriding propagated authorizations when necessary.

A prototype system, called Author-\textit{X}, has been developed to implement the proposed access control model [Bertino et al. 2000]. Author-\textit{X} is built on top of the eXcelon DBMS [Object Design Inc. 1998] as a Java server extension implementing core functionalities of authorization propagation and access control, according to what described in this paper.

We plan to extend the work presented in this paper along several directions. A first direction is related to the extension of the access control model for the aspects related to subjects and protection objects. To support a more flexible authorization specification based on subject qualification and characteristics and not only on subject identity, we intend to introduce a notion of subject credential, as suggested in [Winslett et al. 1997]. With respect to protection objects we are planning to support content-based access control through
the XPath language by [W3C 1999]. A second direction is related to the extension of the Author-XML system. Planned extensions regard the development of optimized access control strategies for authorization evaluation and interactive functionalities for policy selection, to assist the Security Administrator in authorization management according to the results of document classification. Finally, we plan to investigate techniques for XML document view construction, upon access control evaluation.

REFERENCES


