manually assigning an appropriate policy with each document version is, in many cases, a complex and error-prone task. The proposed framework, therefore, incorporates facilities to model interrelationships between authorization and version management and to automate such a complex process.

a. RDF schema for policy and version association.

```
<Policy_Version_Association>
    <policy  rdf:resource="#p1"/>
    <version  rdf:resource="#d1.1"/>
</Policy_Version_Association>
```

b. Example: associating policy p_1 to the document version $d_{1.1}$.

Fig. 4. Policy-version-association schema and example.

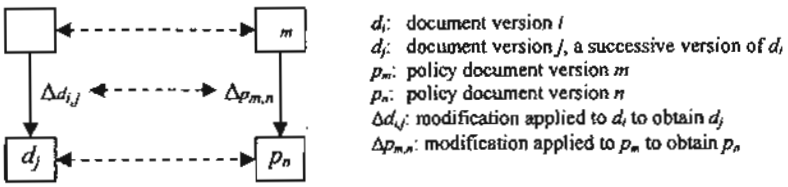


Fig. 5. Access and version control interrelationships.

First, consider the modeled interrelationships of Fig. 5. Assume that authorization of a document version d_i is regulated by a policy p_m. An update $\Delta d_{i,j}$ made to d_i resulting in a newer version d_j may lead to a change of its associated policy, i.e., accesses to d_j will then be regulated by another policy p_n. In particular, there are three possible alternatives to select an appropriate policy for d_j, depending on the type of changes occurred in $\Delta d_{i,j}$:

- A user manually defines an appropriate policy p_n for d_j;
- The system automatically assigns an existing policy p_n, or instantly generates a new policy p_n based on some predefined relationships between $\Delta d_{i,j}$ and $\Delta p_{m,n}$;
- If the above two cases are not applied, by default d_j will be assigned with the same policy as its previous version d_i.

These three methods can be formulated as *RDF clauses* [1], hence allowing the framework to determine the appropriate policy for each version based on some predefined application-specific rules.

Definition 3 (Policy-Version-Association Clause) A policy-version-association clause is represented as an *RDF clause* of the form $H \leftarrow B_1, \ldots, B_n$, where

- H is an RDF expression describing a derived policy-version-association and conforming to the schema of Fig. 4-a,

- B_j is an RDF expression or an RDF constraint restricting a certain condition on the derived association. □

As an example of a policy-version-association clause, consider again the running example of product catalog document. Suppose that in this scenario, the catalog version d_1 is the latest version and is manually associated with the policy p_1 of Fig. 2-a by a security officer. After that d_1 is evolved into $d_{1.1}$ and $d_{1.2}$ with Deltas $\Delta d_{1.1.1}$ and $\Delta d_{1.1.2}$, respectively. The description of version $d_{1.2}$ together with $\Delta d_{1.1.2}$ are given by Fig. 3-b. Since d_1 is evolved into $d_{1.2}$ by updating the value of market-element to "member customers", based on the definition of the policy-version-association clause of Fig. 6-a, the framework can automatically yield an element of Fig. 6-b. That is, the access control of document version $d_{1.2}$ is regulated by the policy p_2.

```
<Policy_Version_Association>
    <policy rdf:resource="#p2"/>
    <version rdf:resource=$S:Y/>
</Policy_Version_Association>
    ← <Version rdf:about=$S:Y>
          <document rdf:resource="catalog.xml"/>
          <previousVersion rdf:resource=$S:X/>
          <delta>
              <xd:xmldiff rdf:parseType="Literal">
                  <xd:change match=$S:anynode>
                      member customers </xd:change>
                  $E:otherchanges
              </xd:xmldiff>
          </delta>
          $E:versionProperties
      </Version>,
      <Policy_Version_Association>
          <policy rdf:resource="#p1"/>
          <version rdf:resource=$S:X/>
      </Policy_Version_Association>
```

a. Policy-version-association clause example.

```
<Policy_Version_Association>
    <policy rdf:resource="#p2"/>
    <version rdf:resource="#d1.2"/>
</Policy_Version_Association>
```

b. Derived Policy_Version_Association element.

```
// This RDF clause formalizes that
// if a document catalog.xml of
// version Y is evolved from ver-
// sion X by changing the value of
// market-element to "member
// customers" and that document
// version X is associated with
// policy p1, then policy p2 will be
// assigned to the new version Y.
//
```

Fig. 6. Policy-version-association clause.

5 Conclusions

This paper has addressed the problems that arise when access and version control of XML documents are handled separately, such as the complexity of designing a single policy for enforcement of authorization over all versions of a document. To deal with these problems, a framework which integrates both access and version control mechanisms has been developed. This unified framework has facilities to handle these two essential mechanisms effectively and simultaneously. Moreover, because access control policies and version information themselves can be represented as XML documents, the framework then provides a complete and uniform solution to easily manipulating schemas, version information and access control policies of XML document databases. Further work includes the following issues:

- incorporation of a mechanism to handle version control over document schemas;

- development of an appropriate technique to solve inference problems (i.e., a situation that allows users to infer certain information about protected data);
- in-depth analysis and refinement of the representation of temporal features to deal with temporal XML document databases and temporal query processing;
- application of the framework to XML data warehouses-repositories of data coming from several sources (e.g., web sites) in which many important issues such as security, consistency and maintenance/querying of history of or changes in document versions are interwoven; and
- implementation of the framework by *RDD's inference engine* [3].

References

1. Anutariya, C., Wuwongse, V., Akama, K. and Nantajeewarawat, E.: RDF Declarative Description: A Language for Metadata. *J. Digital Information*, Vol. 2, Issue 2 (2001).
2. Anutariya, C., Chatvichienchai, S., Iwiahara, M., Wuwongse, V. and Kambayashi, Y.: A Rule-Based XML Access Control Model. *Proc. Rules and Rule Markup Languages for the Semantic Web Workshop*, LNCS #2762, Springer Verlag, pp. 92-103 (October 2003).
3. Anutariya, C., Wuwongse, V. and Wattanapailin, V.: An Equivalent-Transformation-Based XML Rule Language. *Proc. Int'l Workshop Rule Markup Languages for Business Rules in the Semantic Web*, Sardinia, Italy (2002)
4. Bertino, E., Castano, S., Ferrari, S. and Mesiti, M.: Specifying and Enforcing Access Control Policies for XML Document Sources. *World Wide Web*, Vol. 3, No. 3, Baltzer Science Publishers, Netherlands (2000).
5. Chatvichienchai, S., Iwaihara, M. and Kambayashi, Y.: Translating Content-Based Authorizations for XML Documents. *Proc. 4th Int. Conference on Web Information Systems Engineering*, IEEE CS, pp.103–112, Roma, Italy (Dec. 2003).
6. Chatvichienchai, S., Iwaihara, M. and Kambayashi, Y.: Authorization Translation for XML Document Transformation. *J. World Wide Web*, Vol. 7, No. 1, pp. 111-138 (2004).
7. Chien, S. Y., Tsotras, V. J. and Zaniolo, C.: Efficient Management of Multiversion Documents by Object Referencing. *Proc. VLDB 2001* (Sep. 2001).
8. Clark, J. and DeRose, S.: XML Path Language (XPath) Version 1.0 (November 1999). http://www.w3c.org/TR/xpath
9. Cobena, G., Abiteboul, S., Marian, A.: XyDiff Tools Detecting changes in XML Documents. http://wwwrocq.inria.fr/cobena
10. Cuppens, F., Gabillon, A.: Cover story management. *Data Knowledge Engineering*, Vol. 37, No. 2, pp. 177-201 (2001).
11. E. Fernandez, E. Gudes, and H. Song.: A Model for Evaluation and Administration of Security in Object-Oriented Databases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, pp. 275-292 (April 1994).
12. Lassila, O. and Swick, R.R.: Resource Description Framework (RDF) Model and Syntax Specification. *W3C Recommendation* (Feb. 1999) http://www.w3.org/TR/REC-rdf-syntax/
13. Marian, A., Abiteboul, S., Cobna, G. and L. Mignet. Change-centric management of versions in an XML warehouse. *Proc. VLDB 2001* (Sep. 2001).
14. Microsoft XML Diff and Patch 1.0, Microsoft Corporation (2002). http://apps.gotdotnet.com/xmltools/xmldiff/overview.html
15. OASIS XACML Technical Committee: XACML 1.0 Specification Set. *OASIS Standard* (Nov. 2002). http://www.oasis-open.org/committees/xacml
16. Tichy, W. F.: RCS A System for Version Control. *Software Practice and Experience*, Vol. 15, No. 7, pp. 637-654 (July 1985).

(2) ผลงานวิจัยที่ตีพิมพ์ในวารสารวิชาการระดับนานาชาติ

Chutiporn Anutariya, Rachanee Ungrangsi, and Vilas Wuwongse: SQORE: A Framework for Semantic Query Based Ontology Retrieval. *Lecture Notes in Computer Science*, Vol. 4443, ISSN 0302-9743, Springer Verlag, pp. 924 – 929 (April 2007). Impact Factor 0.402

SQORE: A Framework for Semantic Query Based Ontology Retrieval

Chutiporn Anutariya[1], Rachanee Ungrangsi[1], and Vilas Wuwongse[2]

[1] School of Technology, Shinawatra University
99 Moo 10 Bangtoey, Samkok, Pathum Thani, 12160 Thailand
{chutiporn, rachanee}@shinawatra.ac.th
[2] School of Engineering and Technology, Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathum Thani, 12120 Thailand
vw@cs.ait.ac.th

Abstract. Existing approaches to ontology retrieval solely base their search mechanisms on keyword matching while taxonomic structure is solely used for ranking purpose. Users, therefore, are not equipped with expressive means to structurally and semantically describe their ontology needs. To tackle this problem, this paper develops a framework for *Semantic Query based Ontology Retrieval*, namely *SQORE*. It enables precise formulation of a *semantic query* in order to best capture a user' s ontology requirements, which include not only the desired class and property names, but also their relations and restrictions. SQORE employs *XML Declarative Description* (*XDD*) *theory* as its theoretical foundation for modeling ontology databases and evaluating semantic queries. Moreover, *similarity score* is formally defined as a key metric for ranking the resulting ontologies based on their conceptual closeness to the given query.

1 Introduction

The *Semantic Web* aims to expand the World Wide Web by allowing data to be shared and reused across applications and communities via *ontology* [4]. However, existing ontology search engines primarily base their approaches on search terms which cannot sufficiently capture the structural and semantic information about the domain concepts that users want. *Swoogle* [6, 7] and *OntoKhoj* [9] implement the *Google*'s *PageRank*-like algorithm [5] which creates the rankings based on ontology referral network. However, this approach is currently inefficient due to the lack of links among ontologies on the Web. *OntoSearch* [10] provides several criteria for users to evaluate and browse the ontologies and then uses *AKTiveRank* [1] as metrics for ontology ranking based on the taxonomic structure information such as class names, shortest paths, the linking density and the positions of focused classes in the ontology. Although a large number of ontologies are returned as the result in these approaches, they are often un-usable because they do not meet user requirements or otherwise they require tremendous modification efforts. Furthermore, semantic information such as properties and ontological relations (e.g. subClassOf, inverseOf, equivalentClass, and subPropertyOf) is not considered.

R. Kotagiri et al. (Eds.): DASFAA 2007, LNCS 4443, pp. 924 – 929, 2007.

This paper develops *SQORE – Semantic Query based Ontology Retrieval* framework which allows users to structurally and semantically formulate their ontology requirements in terms of *semantic queries*. It employs *XML Declarative Description (XDD) theory* [2, 3, 11] as its theoretical foundation for modeling ontology databases and evaluating semantic queries, which does not only facilitate ontology matching and retrieval, but also support reasoning capability to enhance the matching results. Moreover, it also enables the use of a semantic lexical database, such as *WordNet* [8], for determining semantic relation between two given terms. Thus, the retrieval performance (precision and recall) can be significantly improved when compared to a conventional keyword search. In addition, it employs *similarity score* to rank the resulting ontologies by focusing on their conceptual closeness to the formulated semantic query.

Sect. 2 presents SQORE's overview architecture. Sect. 3 describes ontology database modeling and its semantics. Sect. 4 develops an approach to semantic query formulation and evaluation. Sect. 5 concludes and draws future research directions.

2 SQORE's System Architecture Overview

Fig. 1 illustrates SQORE's system architecture which comprises four major components: i) a *semantic query*, ii) a *retrieval engine*, iii) an *ontology database*, and iv) a *semantic lexical database*. In essence, the system works as follows: First, a user formulates and submits a semantic query which precisely captures his/her ontology requirements. The system then executes such query by semantically evaluating it against the ontology database, which comprises a collection of ontologies and a set of rules defining ontology axiomatic semantics. By incorporating these rules, implicit information about classes/properties in a query and an ontology can be derived, and hence enabling semantic query evaluation. Furthermore, when class/property names defined in a query and an ontology do not *exactly match* (=), four possibilities occur: i) *equivalence* (≡): the two terms are synonym, ii) *more general* (⊇): the query term is broader, iii) *less general* (⊆): the ontology term is broader, and iv) *unknown* (≠): the relation is unknown. To tackle this, a referenced lexical database, such as *WordNet* [8], is employed in order to determine their appropriate semantic relation. Finally, the system computes the *semantic similarity score* between a given query and an ontology in the collection, which ranges from 0 (strong dissimilarity) to 1 (strong similarity), and returns as the answer the list of ranked ontologies.

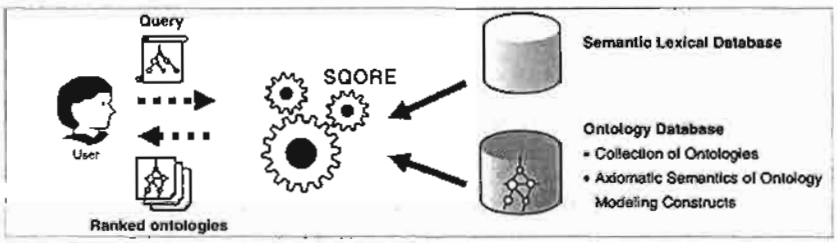


Fig. 1. SQORE System Architecture

3 Ontology Database Model and Its Semantics

By employment of XDD theory, an ontology formalized in RDF(S) or OWL can readily and directly become an XDD description. In order to determine the meaning of a particular ontology, formalization of the axiomatic semantics of each RDF(S)/OWL modeling construct is demanded.

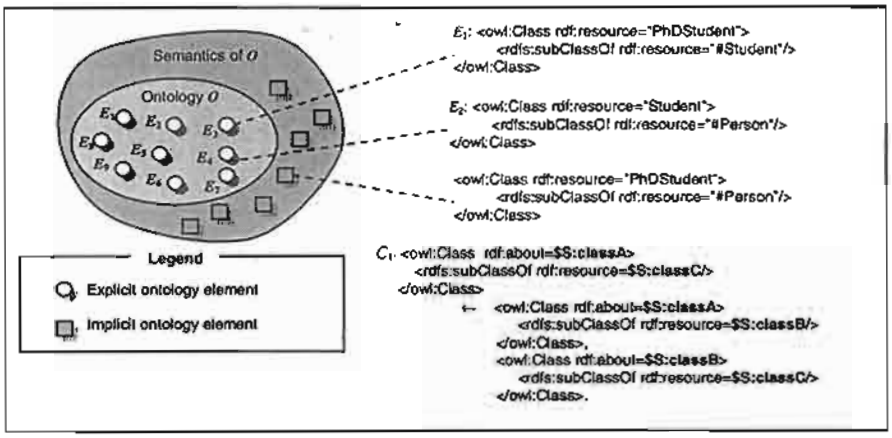


Fig. 2. Semantics of the ontology O wrt. the axiomatic semantics of RDF(S)/OWL constructs

Fig. 2 gives an example of formalizing rdfs:subClassOf in terms of an XDD description C_1. By incorporating this description, the meaning of the ontology O, consequently, yields elements E_1–E_9 which are explicitly defined by O, together with the derived ones. For instance, as depicted by Fig. 2, assume that Query Q searches for ontologies with PhDStudent modeled as a subClassOf Person, while Ontology O defines PhDStudent as a subClassOf Student which is then a subClassOf Person; therefore by modeling the semantics of subClassOf to be transitive, one can derive that in Ontology O, PhDStudent is also a subClassOf Person, and thus satisfying the query requirement.

Founded on this formalism, an ontology database ODB becomes an XDD description with two parts: i) ODB_C: An ontology collection consisting of n ontologies $O_1,\ldots, O_n$, and ii) ODB_A: A set of XML clauses defining the axiomatic semantics of ontology modeling constructs. The database semantics, denoted by $\mathcal{M}(ODB)$, are the set of OWL elements describing classes, properties or instances (individuals) which are directly represented by the database via the ontology collection ODB_C plus those derivable from it via the axiomatic semantics ODB_A.

4 Semantic Query Formulation and Evaluation

Definition 1 (Semantic Query). A *semantic query* is formulated as an XML element of the form:

```
<sq:Query>
  <sq:mandatoryConditions> m1 m2 ... mn </sq:mandatoryConditions>
  <sq:optionalConditions> o1 o2 ... or </sq:optionalConditions>
  <sq:semanticLexiconReference> url </sq:semanticLexiconReference>
  <sq:similarityWeightFactors>
        <sq:mandatoryConditionWeight> wM </sq:mandatoryConditionWeight>
        <sq:stringMatching> w≈ </sq:stringMatching>
        <sq:synonymRelation> w≡ </sq:synonymRelation>
        <sq:moreGeneralRelation> w⊇ </sq:moreGeneralRelation>
        <sq:lessGeneralRelation> w⊆ </sq:lessGeneralRelation>
        <sq:unknownRelation> w∗ </sq:unknownRelation>
  </sq:similarityWeightFactors>
</sq:Query>
```

where

- m_i is an OWL/RDF(S) expression describing a query's mandatory condition,
- o_i is an OWL/RDF(S) expression describing a query's optional condition,
- *url* gives a URL of a semantic lexical database,
- $w_M, w_{\approx}, w_{\equiv}, w_{\supseteq}, w_{\subseteq}, w_{*} \in [0, 1]$ are semantic weight factors,
- semanticLexiconReference- and similarityWeightFactors-elements are optional.

Given a semantic query Q, let $mcond(Q)$ and $ocond(Q)$, respectively, denote the sets of mandatory conditions and optional conditions. □

Formally speaking, an ontology O in ODB satisfies a condition m_i or o_i if such a conditional element is included in the meaning of O. The weight factor w_M allows explicit specification of how important the mandatory conditions are, and hence $1-w_M$ becomes the weight for the optional conditions. The semantic lexicon reference specifies external knowledge used for determining appropriate semantic relations between elements of Q and O. Thus, based on the discovered semantic relation between a query element t_1 and an ontology element t_2 defined as follows, the weight factors $w_{\approx}$, $w_{\equiv}$, $w_{\supseteq}$, $w_{\subseteq}$, w_{*}, respectively, allow the user to quantify how similar the two elements are. In principle, it is recommended that $1 = w_{\approx} \geq w_{\equiv} \geq w_{\supseteq} \geq w_{\subseteq} \geq w_{*} = 0$. In practice, these weights can be configured as default settings of the system or manually defined by a user.

5 Similarity Measures

This section formally defines important similarity measures. First, let Σ be an *ontology alphabet* comprising *ontology elements* in the following sets:

- C: Class names (such as Person, Student, PhDStudent),
- P: Property names (such as firstname, lastname, supervises, isSupervisedBy),
- D: Datatypes (such as xsd:string, xsd:nonNegativeInteger),
- M: Modeling constructs of RDF(S) and OWL (such as rdfs:subClassOf),
- R: Restrictions on classes and properties, having the form $m(a,b)$ where $m \in M$, $a \in (C \cup P)$, $b \in (C \cup P \cup D)$ (such as rdfs:subClassOf(Student, Person)).

The following definition first measures how well two ontology elements match.

Definition 2 (Element Similarity Score: SS_E). The similarity of two ontology elements x and y is measured by:

- For $x,y \in C$ or $x,y \in P$ or $x,y \in D$:

$$SS_E(x,y) = \begin{cases} w_{=} & : \text{if } x = y \\ w_{\equiv} & : \text{if } x \equiv y \\ w_{\supseteq} & : \text{if } x \supseteq y \\ w_{\subseteq} & : \text{if } x \subseteq y \\ w_{\times} & : \text{otherwise} \end{cases} \quad (1)$$

- For $x = m(a_1,b_1) \in R$ and $y = m(a_2,b_2) \in R$:

$$SS_E(x,y) = SS_E(a_1,a_2) * SS_E(b_1,b_2) \quad (2)$$

- Otherwise:

$$SS_E(x,y) = \text{unknown} \quad (3)$$

□

Based on this element similarity score definition, one can measure the similarity between a given element x in Σ and an ontology O in ODB by finding the highest similarity score between x and each element y that is semantically defined by O. This is defined by:

Definition 3 (Best Element Similarity Score: SS_B). The similarity between an element $x \in C \cup P \cup R$ and an ontology $O \in ODB$ is measured by:

$$SS_B(x,O) = \max_{y \in M(O)} S(x,y) \quad (4)$$

□

Next, the satisfaction scores of mandatory conditions (SS_M) and of optional conditions (SS_O) are defined.

Definition 4 (Satisfaction Scores of Mandatory conditions: SS_M and Optional conditions: SS_O). With respect to an ontology O, the satisfaction scores of Q's mandatory conditions (SS_M) and Q's optional conditions (SS_O) are measured by:

$$SS_M(Q,O) = \begin{cases} 1 & : \text{if } mcond(Q) = \varnothing \\ 0 & : \text{if } \exists (c \in mcond(Q))\ SS_B(c,O) \leq w_{\times} \\ \dfrac{\sum_{c \in mcond(Q)} SS_B(c,O)}{|mcond(Q)|} & : \text{otherwise} \end{cases} \quad (5)$$

$$SS_O(Q,O) = \begin{cases} 1 & : \text{if } ocond(Q) = \varnothing \\ 0 & : \text{if } SS_M(Q,O) = 0 \\ \dfrac{\sum_{c \in ocond(Q)} SS_B(c,O)}{|ocond(Q)|} & : \text{otherwise} \end{cases} \quad (6)$$

□

Finally, one can measure the similarity between a semantic query Q and an ontology O by integrating the two components: mandatory condition satisfaction score and optional condition satisfaction score, as follows:

Definition 5 (Query-Ontology Similarity Score). The similarity between a semantic query Q and an ontology O is measured by:

$$SS(Q,O) = w_M SS_M(mcond_Q, O) + (1 - w_M) SS_O(ocond_Q, O) \quad (7)$$

□

6 Conclusions

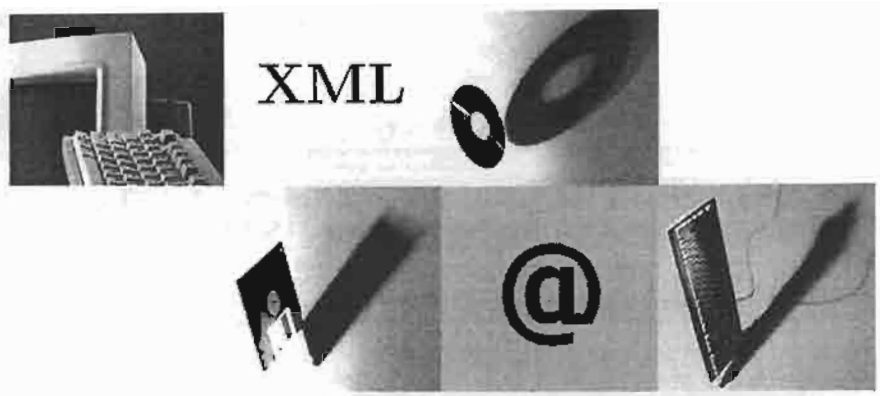
This paper proposes and develops *SQORE* – a novel framework for ontology retrieval system based on *semantic query*. It enables a user to precisely and structurally formulate their ontology requirements, which include not only the desired class and property names, but also their relations and restrictions. Moreover, when evaluating a query, its semantics together with an ontology's semantics are also taken into account in order to correctly and semantically match them.

Acknowledgement

This work was supported by Thailand Research Fund and Commission on Higher Education, Thailand, under grant number MRG4780192.

References

1. Alani, H. and Brewster, C.: Metrics for Ranking Ontologies. In Proceedings of 4th International EON Workshop, 15th Int'l WWW Conference, Edinburgh, 2006.
2. Anutariya, C., Wuwongse, V. and Akama, K.: XML Declarative Description with First-Order Logical Constraints. Computational Intelligence, Vol. 21, No. 2, pp. 130-156 (2005)
3. Anutariya, C., Wuwongse, V., Akama, K., Wattanapailin, V.: Semantic Web Modeling and Programming with XDD. Proc. Semantic Web Working Symposium (SWWS-01), CA (2001), 161–180.
4. Berners-Lee, T., Handler, J., and Lassila, O.: The Semantic Web, Scientific American, May 2001.
5. Brin, S., and Page, L. The anatomy of a large-scale hyper-textual web search engine. In Seventh International WorldWide Web Conference, Brisbane, Australia, 1998
6. Ding, L., Finin, T., Joshi, A., Pan, R., Scott Cost, R., Peng, Y., Reddivari, P., Doshi, V., Sachs, J., Swoogle: a search and metadata engine for the semantic web. Proc. 13 ACM Int'l Conf. Information and Knowledge Management, November 08-13, 2004, DC
7. Finin, T., Mayfield, J., Joshi, A., Scott Cost, R., Fink, C.: Information Retrieval and the Semantic Web, Proc. 38th Annual Hawaii Int'l Conf. System Sciences (HICSS'05) - Track 4, p.113.1, 2005
8. Miller, A., Wordnet: A lexical database for English. In Communications of the ACM, number 38(11), 1995.
9. Patel, C., Supekar, K., Lee, Y., Park, E. K., OntoKhoj: a semantic web portal for ontology searching, ranking and classification, Proc. 5th ACM Int'l Workshop on Web Information and Data Management, November 07-08, 2003, Louisiana.
10. Thomas, E., Alani, H., Sleeman, D. and Brewster, C.: Searching and Ranking Ontologies on the Semantic Web. In Proc. Workshop Ontology Management: Searching, Selection, Ranking, and Segmentation. 3rd K-CAP 2005, pp. 57-60, Banff, Canada
11. Wuwongse, V., Anutariya, C., Akama, K., Nantajeewarawat, E.: XML Declarative Description (XDD): A Language for the Semantic Web. IEEE Intelligent Systems, Vol. 16, No. 3, pp. 54–65 (May/June 2001)



Towards Integration of XML Document Access and Version Control

Authors

- S. Chatvichienchai (Siebold University of Nagasaki, Japan)
- C.Anutariya (Shinawatra U., Thailand),
- V.Wuwongse (Asian Institute of Technology, Thailand),
- M.Iwaihara (Kyoto U., Japan)
- Y.Kambayashi (Kyoto U., Japan) passed away on February 6th, 2004.

Contribution

- Investigate interaction problems of access and version control of XML documents
- Propose a novel framework integrating access and version control to solve the problems
- Formalize their interrelationships by means of RDF Declarative Description (RDD) Theory

Outline of presentation

- Background
  - Access control for XML documents
  - Version control for XML documents
- Requirements and issues of integrating access and version control
- Proposed framework that formalizes the inter-relationships of access and version control by RDF Description
- Conclusion and future work

Access Control on XML Documents

- Application of XML Documents : Electronic Catalogs, Invoices, Contacts, Patient Records etc
- Access control on shared XML documents is essential

Product Catalog

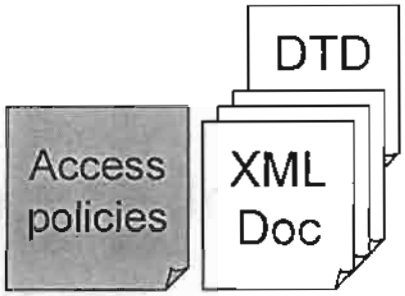
```
<product id="P01234">
  <name> Green Tea 250g </name>
  <price> 1,500 </price>
  <cost> 700 </cost>
</product>
```

Granularity of access control: element level of XML documents

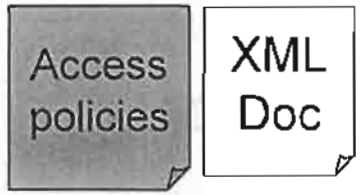
- Sales managers are allowed to read all data of products.
- Customers are not allowed to read costs of products

Related work of XML access control

- Proposed models: AuthorX (E.Bertino et.al [WWW'00]), Rule-Based XML Access Control Model (C.Anutariya et.al [RuleML'03]), and XACML of OASIS etc.
- Common properties
  - Fined-grained access control: elements of document
  - Coarse-grained access control: the whole document



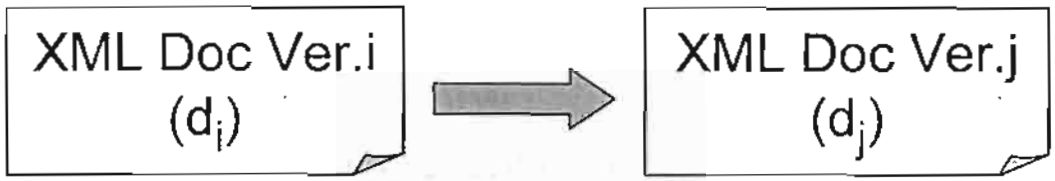
Definition of user access rights for a set of schema instances

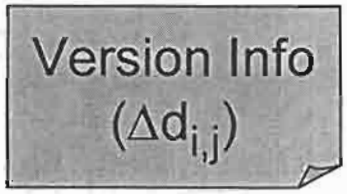


Definition of user access rights for a specific document

Version control for XML documents

- Keep track of changes of evolving XML documents
- Maintain appropriate version information





- users who modify the documents
- Date / time of modification
- modification details

- Enabling the documents to be reverted back to their previous versions

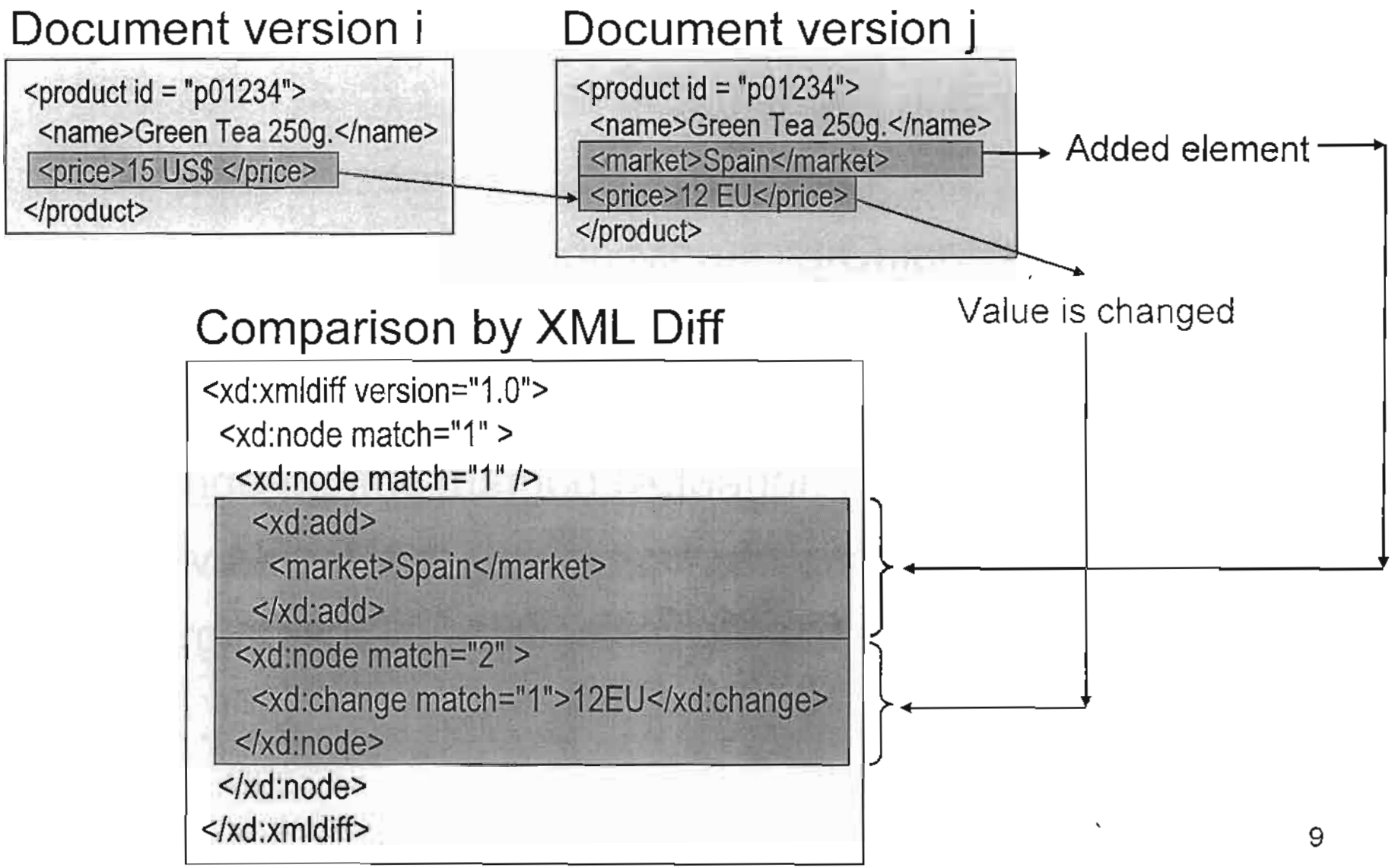
Example: Given d_j and $\Delta d_{i,j}$, we can compute d_i.

How to compute Delta $\Delta d_{i,j}$ of XML document

Given a pair of documents d_i and d_j, delta $\Delta d_{i,j}$ can be computed by one of the following methods:

- a script generated directly from the edit commands of a structured editor.
- Document comparison analyzed by a structured diff package, such as
  - XyDiff Tool (http://wwwrocq.inria.fr/cobena), or
  - Microsoft XML Diff (http://apps.gotdotnet.com/xmltools/xmldiff/overview.html).

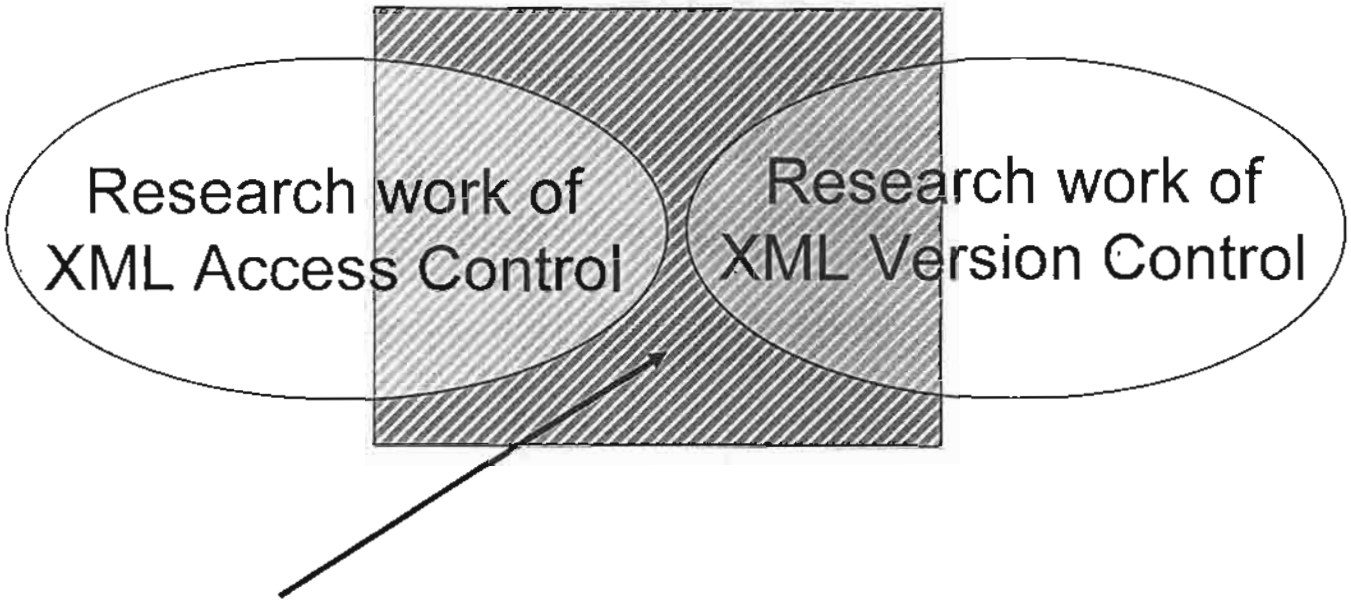
Document comparison by XML Diff



Related work of version control

- In a typical Revision Control System (RCS) scheme (W. F. Tichy [SPE'85]), changes of document d_i to d_j are stored in a script denoted by delta $\Delta d_{i,j}$.
- Reference-Based Version Model (S.Chien et.al [VLDB01]) focuses on the storage performance of multi-version XML documents.
- A change-centric method (A.Marian et.al[VLDB02]) stores the latest version of a XML document and the sequence of forward completed deltas.

Open Problems



There is no work discussing interaction problems between access and version control of XML documents.

Integration need of access & version control (1/2)

1. Version control simplifies schema design and access control policy definition.

Catalog.xml

```
<product id = "p01234">
  <name>Green Tea 250g.</name>
  <price>15 US$ </price>
</product>
```

Catalog.xml

```
<product id = "p01234">
  <market country="USA">
    <name>Green Tea 250g.</name>
    <price>15 US$ </price>
  </market>
  <market country="Spain">
    <name>Té verde 250g.</name>
    <price>12 EU</price>
  </market>
</product>
```

In case without version control

Catalog.xml

Ver.0

```
<product id = "p01234">
  <name>Green Tea 250g.</name>
  <price>15 US$ </price>
</product>
```

Ver.1

```
<product id = "p01234">
  <name>Green Tea 250g.</name>
  <market>USA</market>
  <price>15 US$ </price>
</product>
```

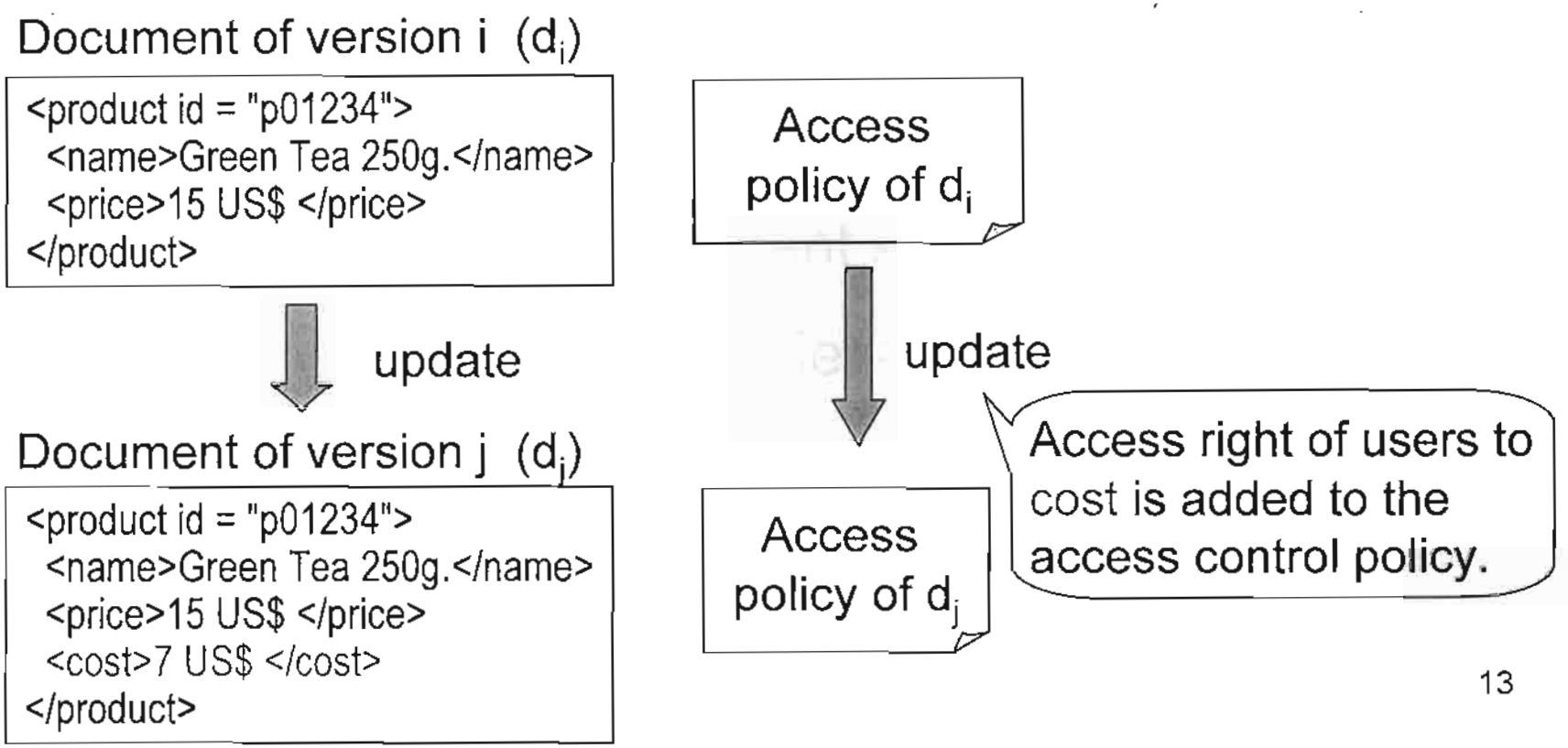
Ver.2

```
<product id = "p01234">
  <name>Té verde 250g.</name>
  <market>España</market>
  <price>12 EU</price>
</product>
```

In case with version control

Integration need of access & version control (2/2)

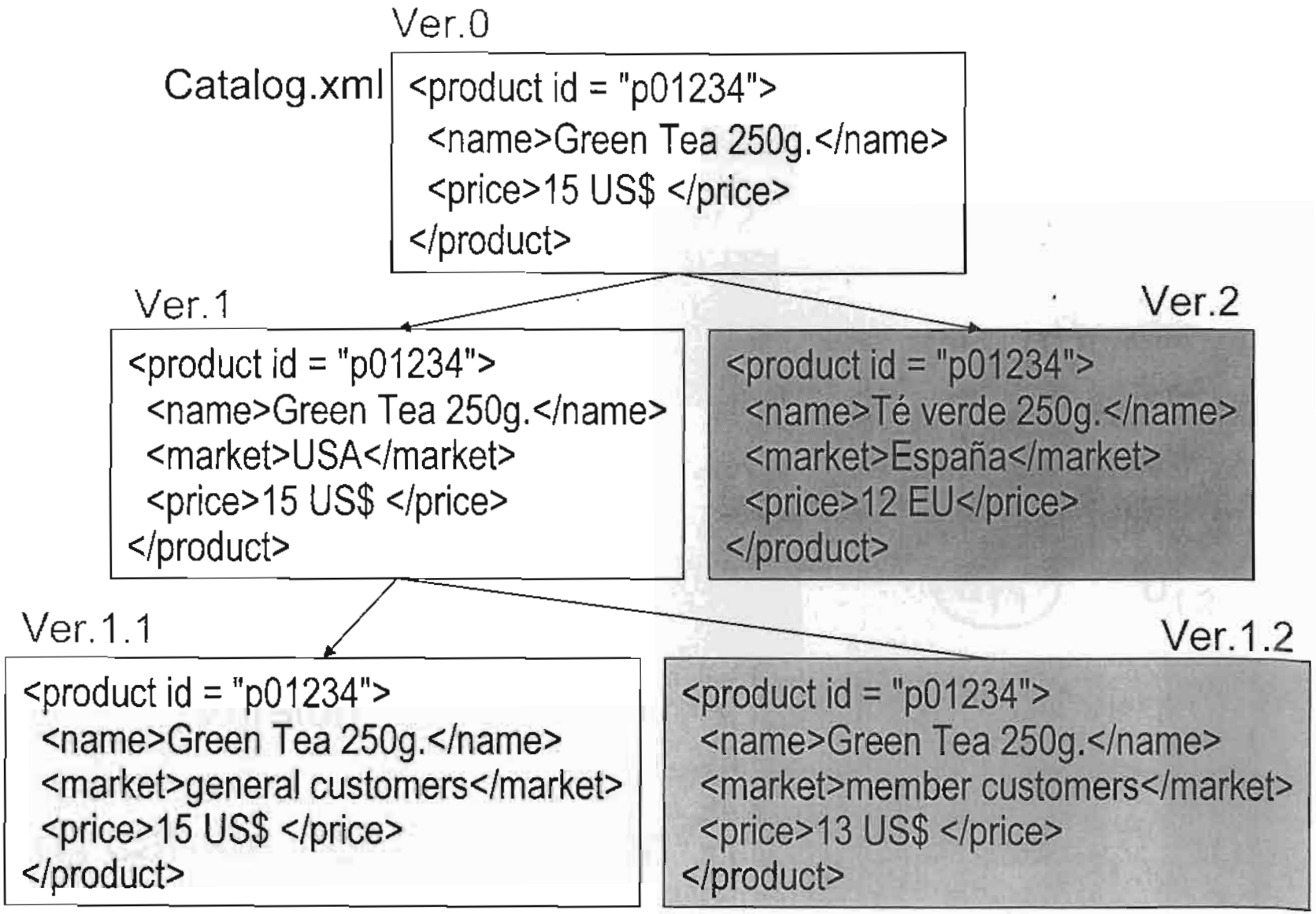
1. Version control simplifies schema design and access control policy definition.
2. Define an access policy of a document of new version based on that of document of current version.



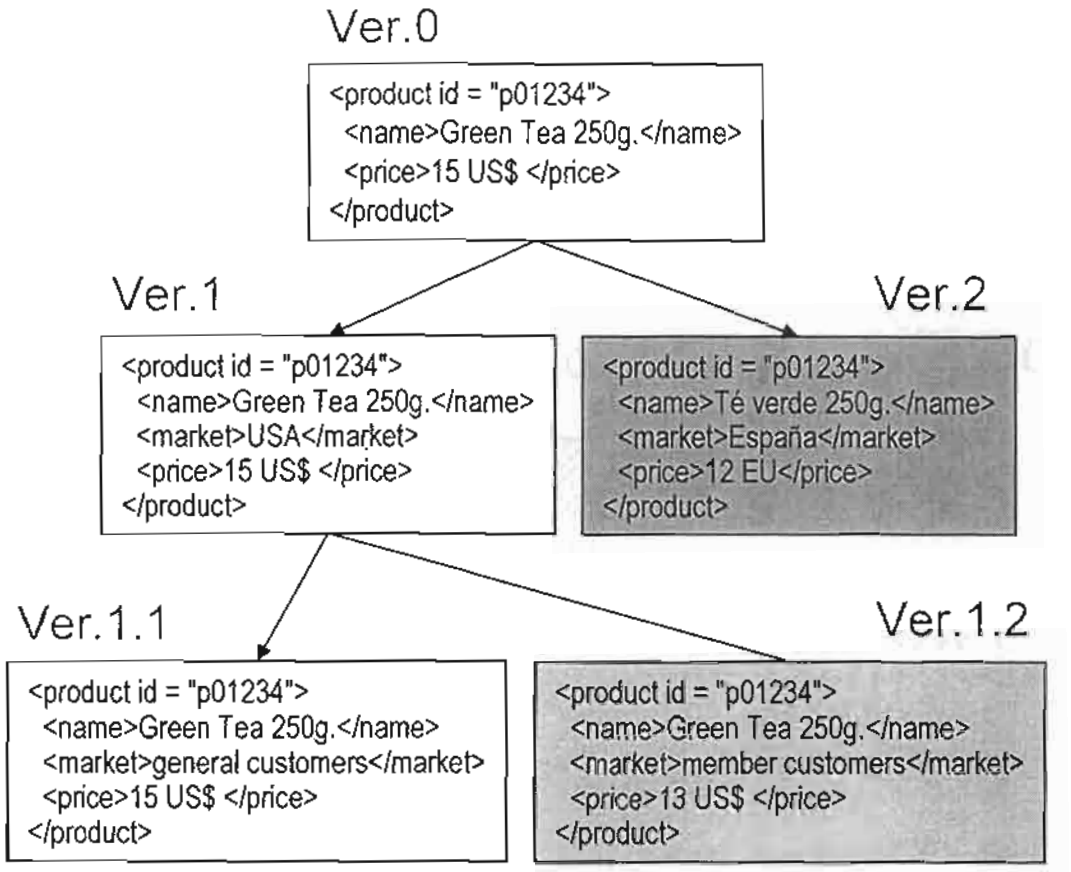
Issues on access and version control integration

- Inference problems
  - Existence of sensitive documents by version history
  - Sensitive data contained in documents of new version
- Improper disclosure of confidential data from updating document
- Version control over document schemas
- Temporal features allowing definition of access control policies with certain conditions on timestamp attributes

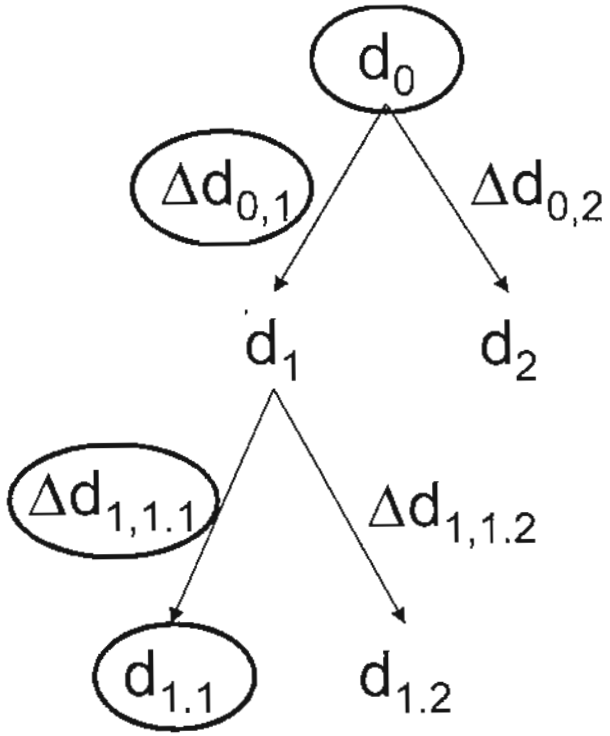
Version history



Representation of version history by version tree



Version history

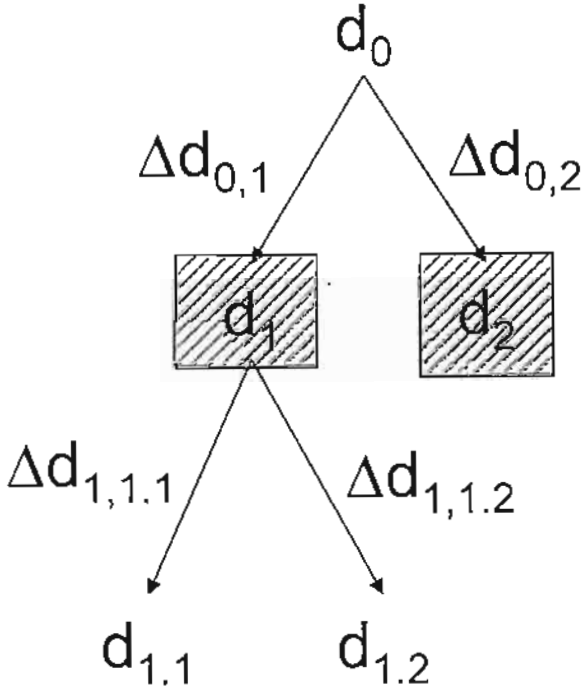


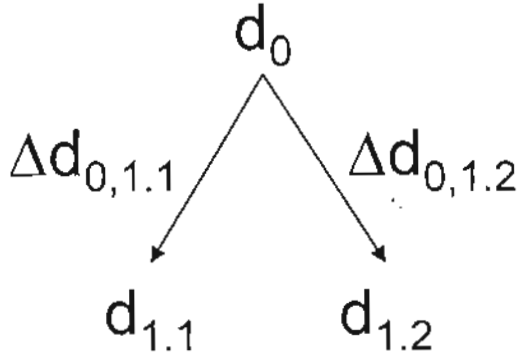
Proposed version tree

d_0 can be computed from given $d_{1.1}$, $\Delta d_{1,1,1}$, $\Delta d_{0,1}$

Inferring existence of sensitive documents by version history

Assume that Bob is not allowed to access documents of ver.1 and 2.

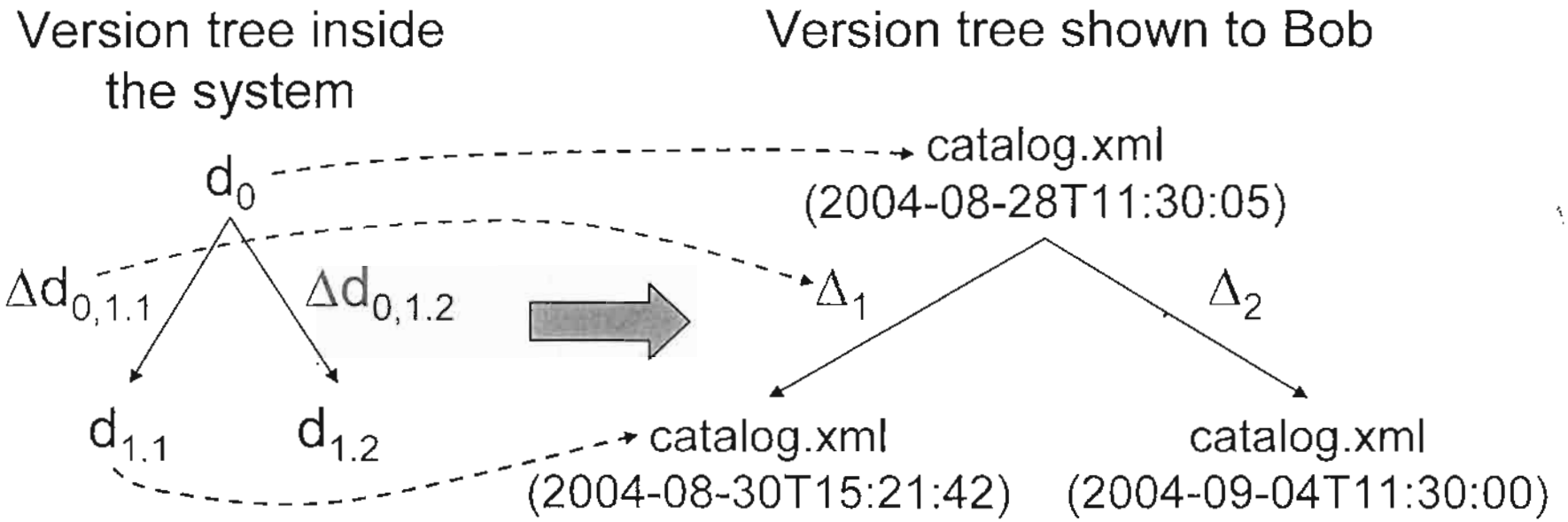




View of version tree seen by Bob.
(d_1 does not appear between d_0 and $d_{1.1}$).

Problem: Bob can aware of d_1 if its filename denotes the disappeared version that is between d_0 and $d_{1.1}$.

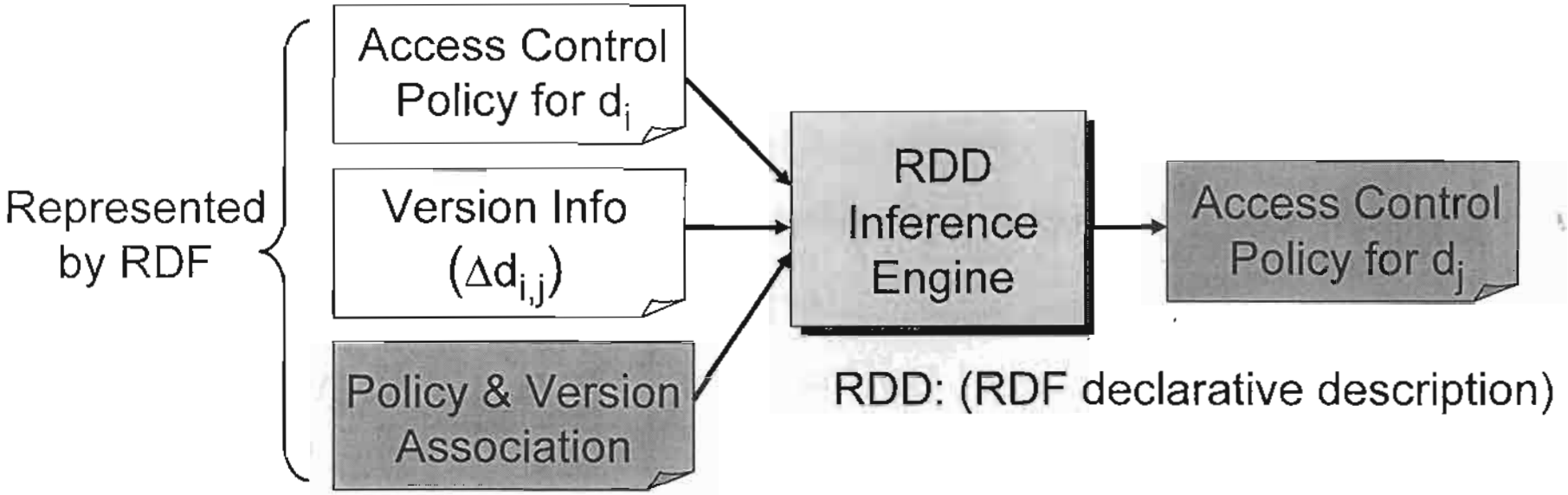
Solution of Inferring existence of sensitive documents



Solution:

1. d_i is represented by its file name that is based on timestamp.
2. $\Delta d_{i,j}$ is represented by Δ_k where k is an integer.

Proposed framework



RDF (Resource Description Framework) is developed by the W3C for Web-based metadata, using XML as an interchange syntax.

RDF allows users defining metadata that has scalability and flexible structure better than by those defined by XML.

RDF (Resource Description Framework)

RDF is built on the following rules:

- a *Resource* is anything that can have a URI; this includes all the world's Web pages, as well as individual elements of an XML document.
- a *PropertyType* is a resource that has a name and can be used as a property, for example Author or Title.
- a *Property* is the combination of a Resource, a PropertyType, and a value.

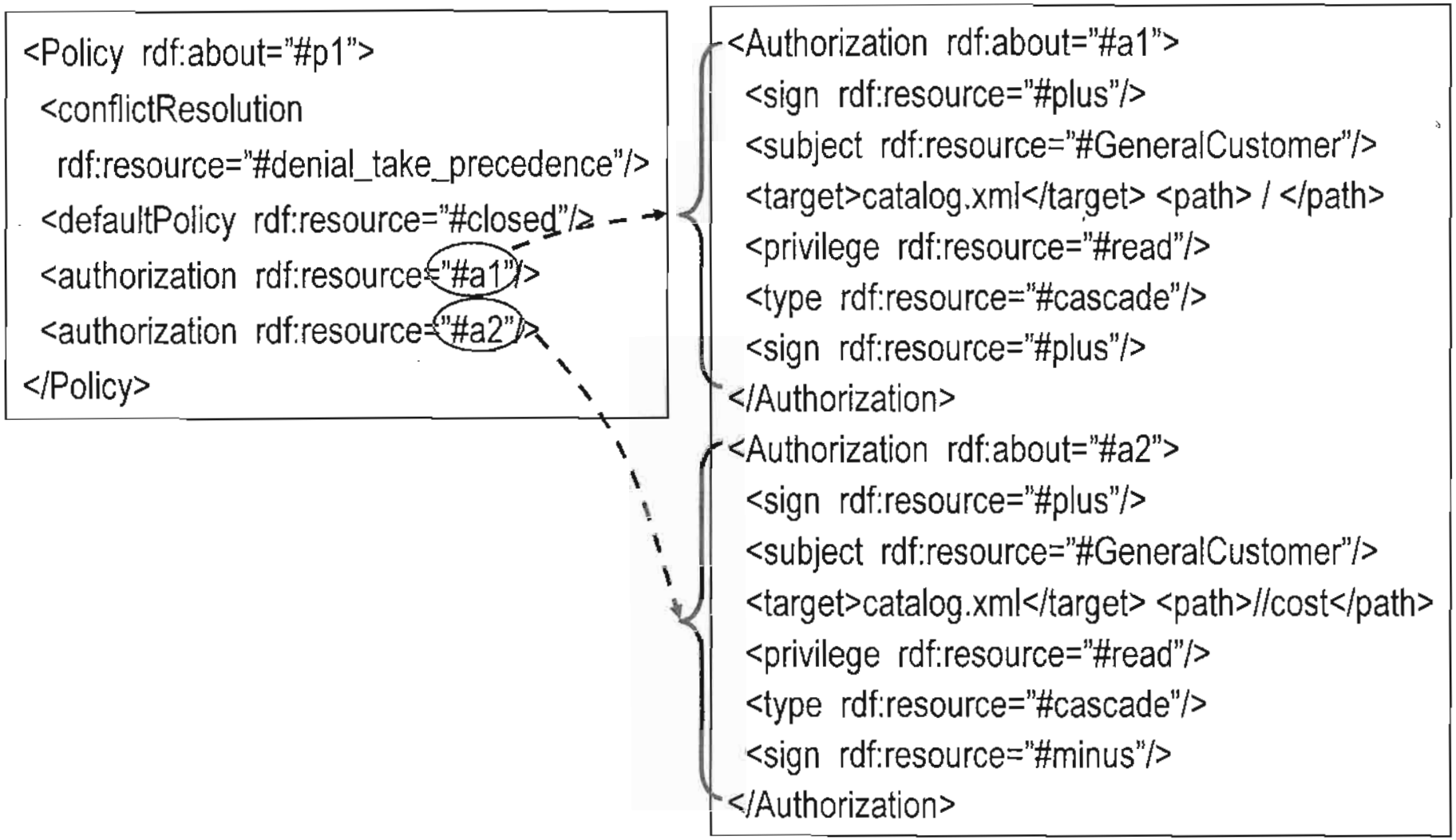
Access control policies

Definition of an *access control policy* consists of the following arguments:

- Subject: a user, user group or a role,
- Target: an XML document,
- Path: an XPath expression identifying data in the document,
- Level: instance level or schema level,
- Privilege: read or write operation,
- Type: cascade or no_propagation,
- Sign: '+'(grant) or '-'(deny),
- Priority: conflict resolution policy (denial_take_precedence, descendant_take_precedence etc),
- Default: open or close policy.

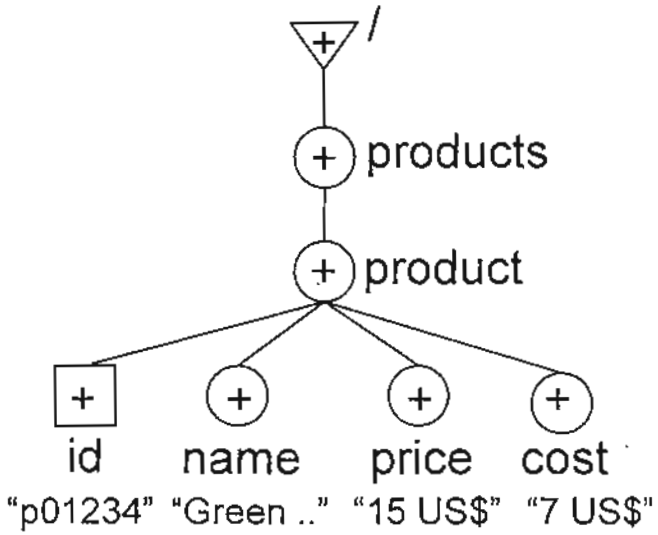
Access control policy defined by RDF (1/4)

An access policy p1 consists of two authorization rules a1 and a2.



Access control policy defined by RDF (2/4)

Based on a1, access right of Bob to each node is granted.

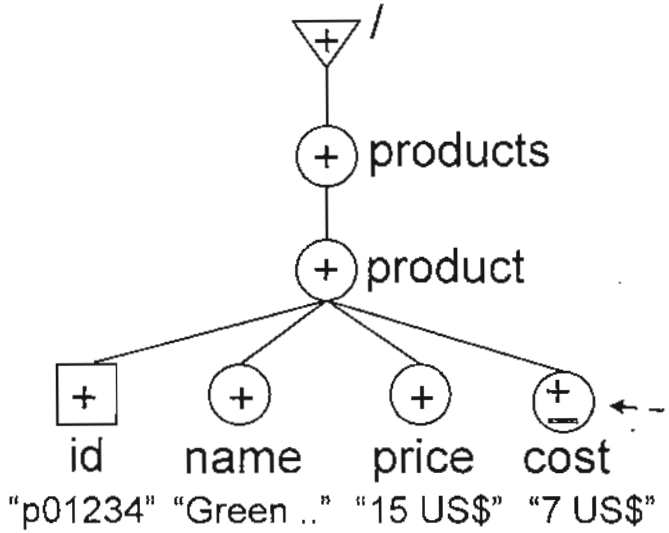


Document tree of catalog.xml

```
<Authorization rdf:about="#a1">
  <sign rdf:resource="#plus"/>
  <subject rdf:resource="#Bob"/>
  <target>catalog.xml</target> <path> / </path>
  <privilege rdf:resource="#read"/>
  <type rdf:resource="#cascade"/>
  <sign rdf:resource="#plus"/>
</Authorization>
<Authorization rdf:about="#a2">
  <sign rdf:resource="#plus"/>
  <subject rdf:resource="#GeneralCustomer"/>
  <target>catalog.xml</target> <path>//cost</path>
  <privilege rdf:resource="#read"/>
  <type rdf:resource="#cascade"/>
  <sign rdf:resource="#minus"/>
</Authorization>
```

Access control policy defined by RDF (3/4)

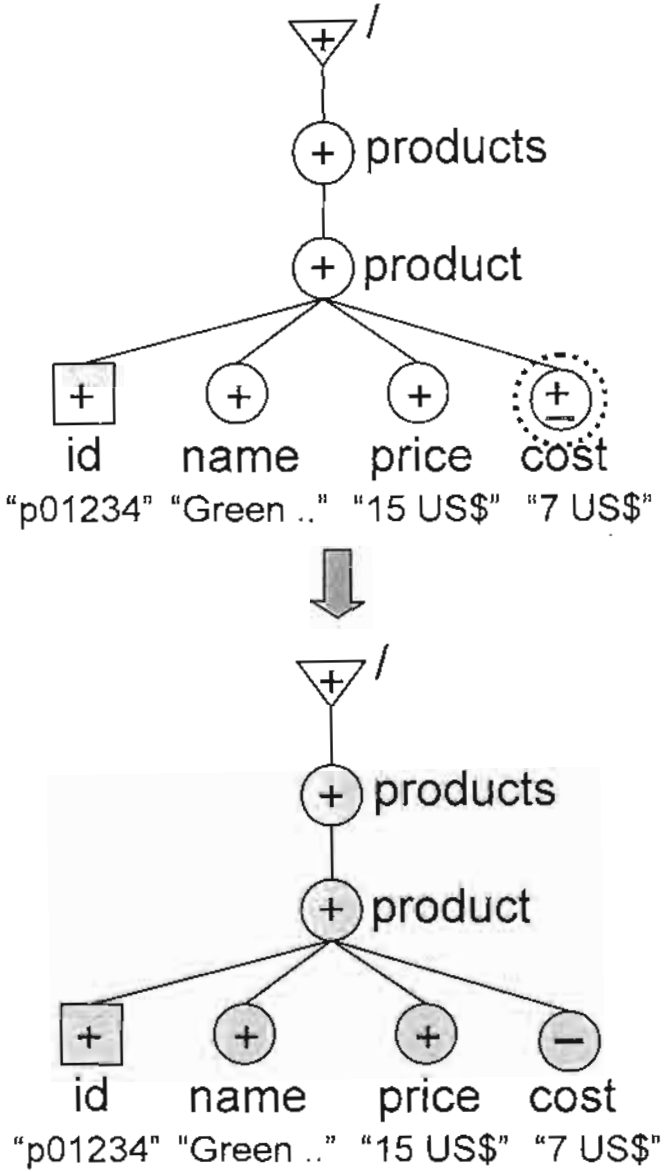
Based on authorization rule a1, Bob is allowed read each node.



Document tree of catalog.xml

```
<Authorization rdf:about="#a1">
  <sign rdf:resource="#plus"/>
  <subject rdf:resource="#Bob"/>
  <target>catalog.xml</target> <path> / </path>
  <privilege rdf:resource="#read"/>
  <type rdf:resource="#cascade"/>
  <sign rdf:resource="#plus"/>
</Authorization>
<Authorization rdf:about="#a2">
  <sign rdf:resource="#plus"/>
  <subject rdf:resource="#Bob"/>
  <target>catalog.xml</target> <path>//cost</path>
  <privilege rdf:resource="#read"/>
  <type rdf:resource="#cascade"/>
  <sign rdf:resource="#minus"/>
</Authorization>
```

Access control policy defined by RDF (4/4)




```
<Policy rdf:about="#p1">
  <conflictResolution
    rdf:resource="#denial_take_precedence"/>
  <defaultPolicy rdf:resource="#closed"/>
  <authorization rdf:resource="#a1"/>
  <authorization rdf:resource="#a2"/>
</Policy>
```

Version information

Definition of version information consists of the following arguments:

- Document: a reference to the baseline version of a document;
- PreviousVersion: a reference to its previous version,
- Creator: author of the described version,
- TimeStamp: the version's timestamp,
- ValidFrom, ValidTo: representing valid time interval of the described version,
- Delta: changes between the previous and current versions specified in XML Diff language.

Version information defined by RDF

```
<Version rdf:about="#d1.2">
 <document rdf:resource="catalog.xml"/>
 <previousVersion rdf:resource="#d1"/>
 <creator rdf:resource="#john"/>
 <timestamp>2004-02-01T09:00:00</timestamp>
 <validFrom>2004-03-01T00:00:00</validFrom>
 <validTo>2004-04-01T00:00:00</validTo>
  <delta rdf:parseType ="Literal">
   <xd:xmldiff>
    <xd:node match="1">
     <xd:change match="2">member customers</xd:change>
     <xd:change match="3">1300Yen</xd:change>
    </xd:node>
   </xd:xmldiff>
  </delta>
</Version>
```

Policy-version-association

A policy-version-association clause is represented as an RDF clause of the following form:

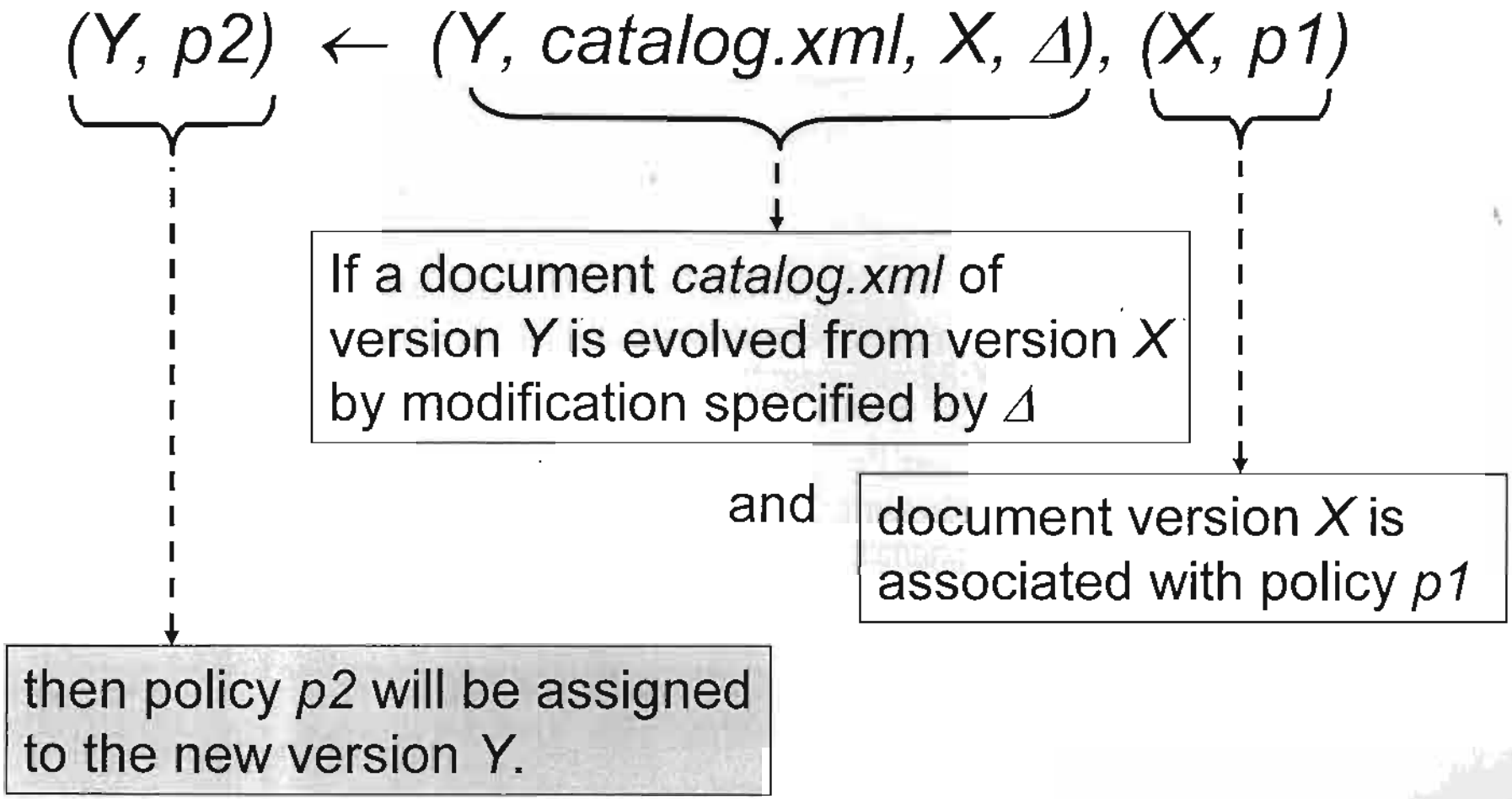
$$H \leftarrow B_1, \ldots, B_n,$$

where

H: an RDF expression describing a derived policy-version-association,

B_i : an RDF expression or an RDF constraint restricting a certain condition on the derived association.

A sample of policy-version-association



Policy-version-association described by RDF

(Y, p2) ← (Y, catalog.xml, X, Δ), (X, p1)

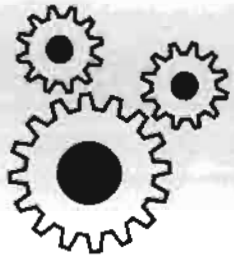
```
<Policy_Version_Association>
 <policy rdf:resource="#p2"/>
 <version rdf:resource=$S:Y/>
</Policy_Version_Association>
  ← <Version rdf:about=$S:Y>
        <document rdf:resource="catalog.xml"/>
        <previousVersion rdf:resource=$S:X/>
        <delta>
         <xd:xmldiff rdf:parseType="Literal">
          <xd:change match=$S:anynode>
            member customers </xd:change>
                $E:otherchanges
         </xd:xmldiff>
        </delta>
        $E:versionProperties
    </Version>,
    <Policy_Version_Association>
     <policy rdf:resource="#p1"/>
        <version rdf:resource=$S:X/>
    </Policy_Version_Association>
```

Conclusions

- Investigated interaction problems of access and version control of XML documents
- Proposed a novel framework integrating access and version control to solve the problems
- Formalized their interrelationships by means of RDF Declarative Description Theory

Further work

1. Implementation of the framework by RDD's inference engine
2. A mechanism that handles version control over document schemas
3. Detection of document modification (Δ) that results in improper disclosure of confidential data
4. Temporal features that deal with temporal XML document DB and temporal query processing



SQORE

A framework for Semantic Query based Ontology REtrieval

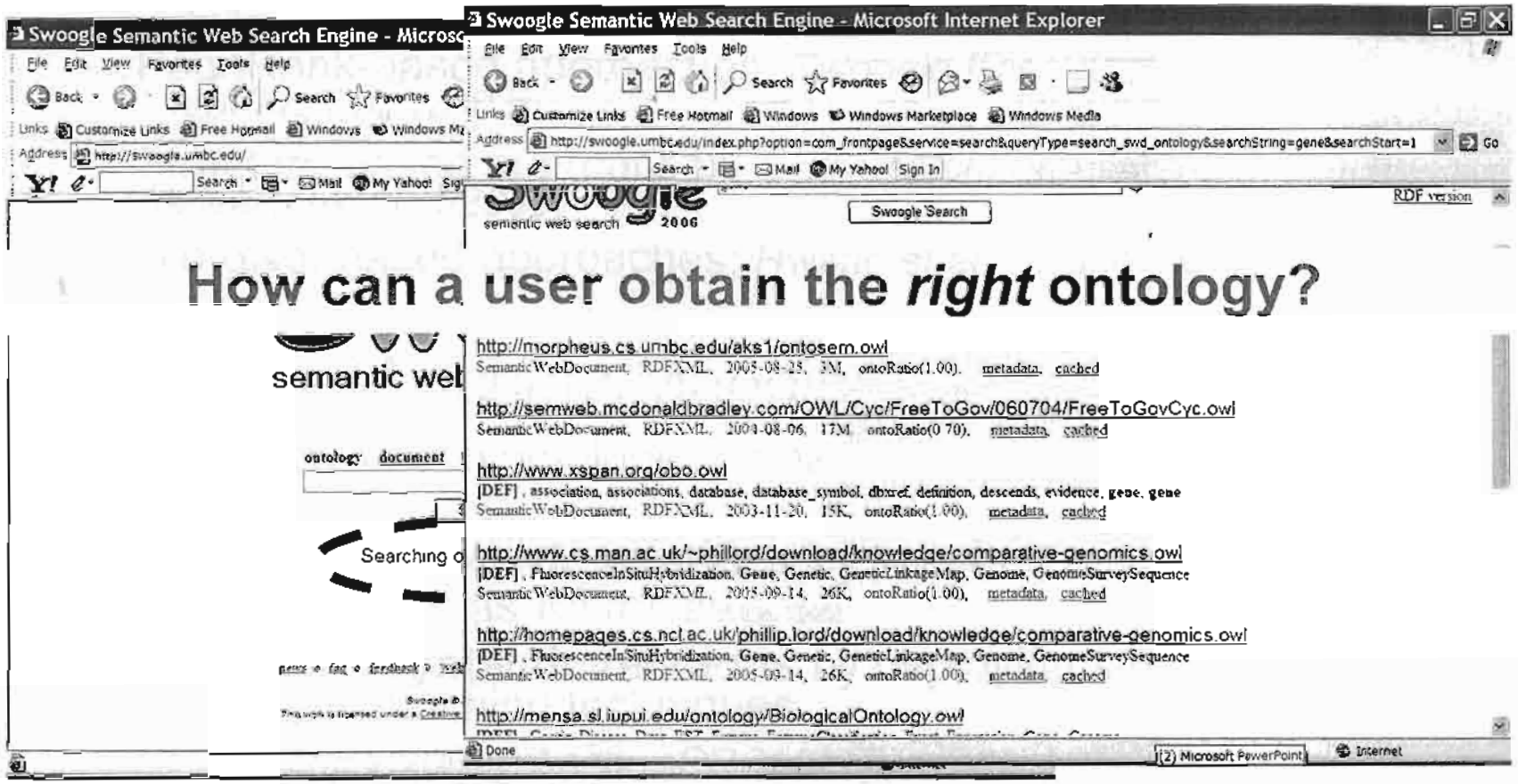
Chutiporn Anutariya[1] , *Rachanee Ungrangsi[1]*, Vilas Wuwongse[2]
Shinawatra University[1], Asian Institute of Technology[2]

This work was supported by Thailand Research Fund and Commission on Higher Education, Thailand, under grant number MRG4780192.

Outline

- Motivation
- Literature Review
- SQORE System Architecture
- Similarity Measures
- An Example
- Conclusions

Motivation: Ontology Reuse Problem



Literature Review

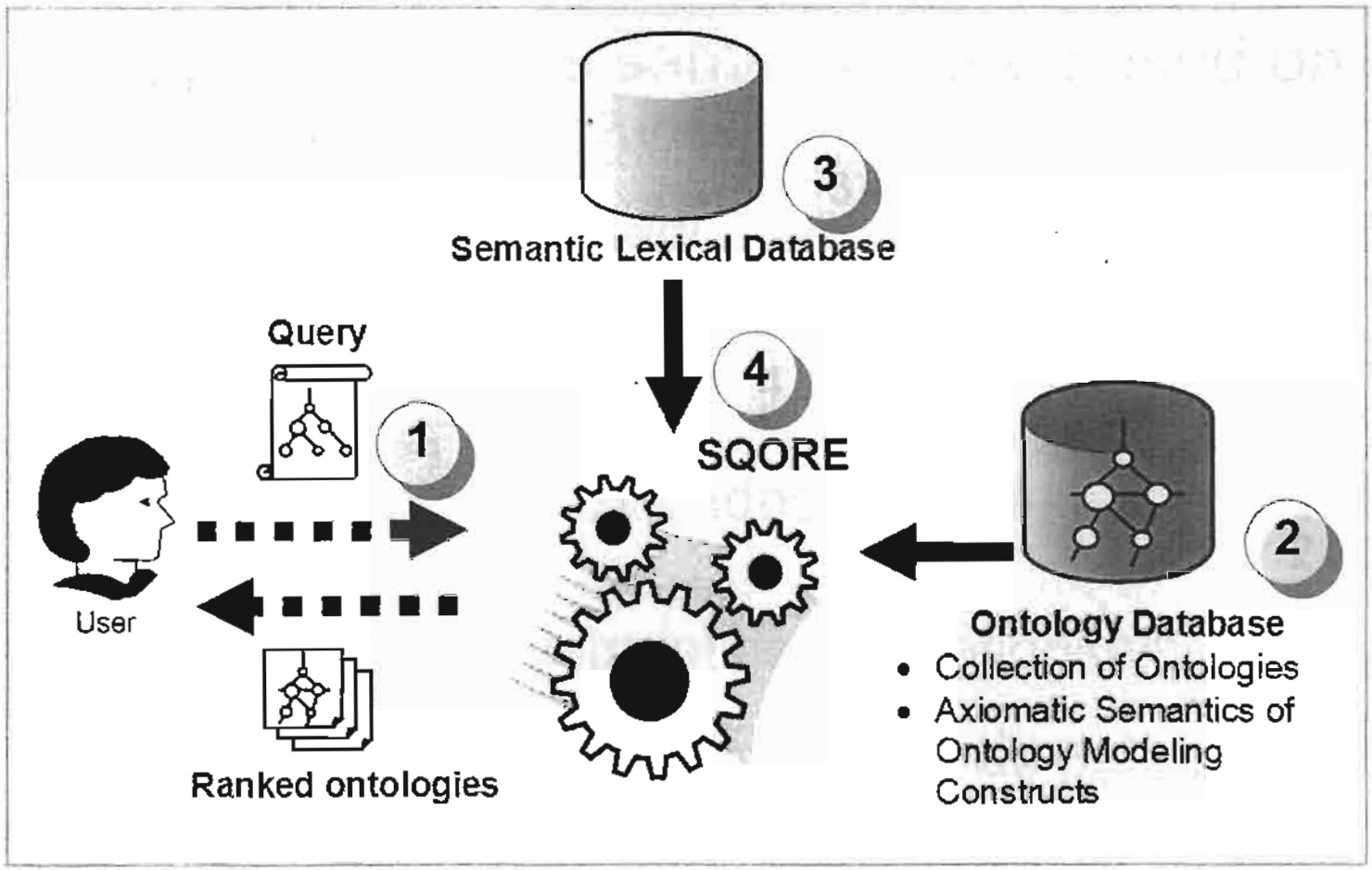
- Ontology Retrieval
  - PageRank-based approaches: *Swoogle [Ding2004], OntoKhoj[Patel2003]*
  - Structure-based approaches: *OntoSearch[Thomas2005], AktiveRank[Alani2006]*
  - WordNet-based approaches: *Hwang et.al [Hwang2006]*

- Ontology Matching/Mapping
  - **Similarity** of two ontologies is based on semantic and syntactic relations
  - determines **semantic relations** by referring to linguistic resources such as *WordNet*[Miller1995]
  - determines **syntactic relations** by deploying *string* matching and *graph* matching techniques
  - *i.e. COMA[Do2002], Anchor-PROMPT[Noy2001], S-Match[Giunchiglia2004]*

Drawbacks

- Keywords cannot capture user's ontology requirements efficiently.
- Existing Ontology Retrieval systems ignore Axiomatic Semantic information such as properties and ontological relations
- Using only WordNet to determine semantics is insufficient

SQORE System Architecture



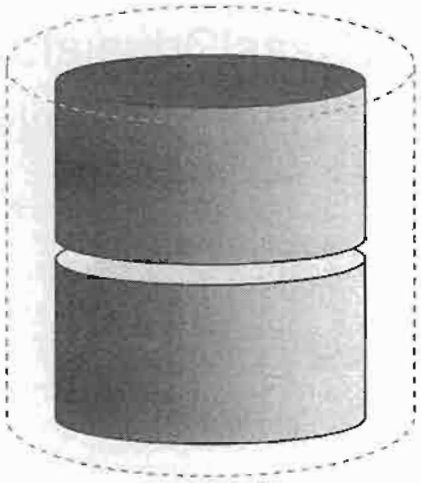
Ontology Database Model

- ODB model and its semantics are based on *XML Declarative Description (XDD) theory* *[Anutariya01, Wuwongse01]*
- A description in XDD is a set of
  - *Ordinary XML elements*
  - *XML expressions*: extended XML elements with variables

    : constraints and relationships
- OWL or RDF/RDFS ontologies directly becomes an XDD description

Ontology Database Model

- An ontology database *ODB* becomes an XDD description with two parts:



$ODB_{Collection\ of\ Ontologies}$

$ODB_{Axiomatic\ Semantics}$

- By incorporating ODB_A, *implicit information* about classes/properties in a query and ontology *can be derived*, and hence enabling semantic query evaluation.

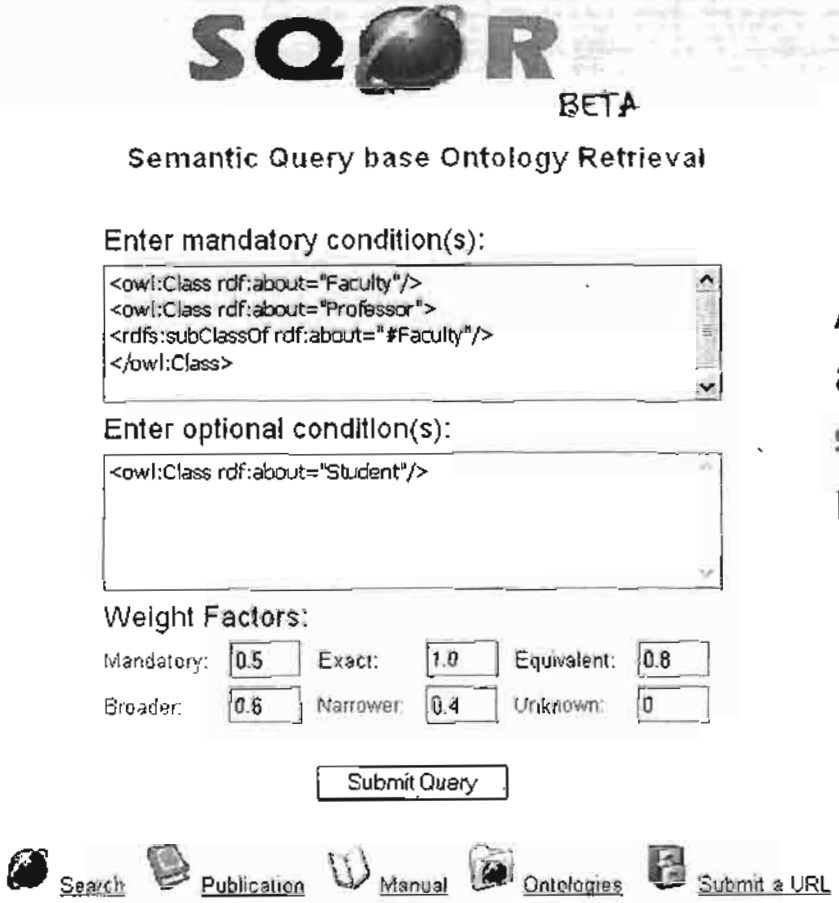
Ontology Database Model: an Example

```
C1: <owl:Class  rdf:about=$S:classA>
        <rdfs:subClassOf rdf:resource=$S:classC/>
    </owl:Class>
        ←   <owl:Class rdf:about=$S:classA>
                <rdfs:subClassOf rdf:resource=$S:classB/>
            </owl:Class>,
            <owl:Class rdf:about=$S:classB>
                <rdfs:subClassOf rdf:resource=$S:classC/>
            </owl:Class>.
```

Semantic Lexical Database

- When class/property names defined in a query and an ontology do not match (=), four possible semantic relations occur:
  - i) : the two terms are synonym,
  - ii) *more general*($\supseteq$): the query term is broader,
  - iii) *less general*($\subseteq$): the ontology term is broader, and
  - iv) *unknown*($\neq$): the relation is unknown.
- Semantic lexical database such as *WordNet*[Miller1995] is employed to determine semantic relations

Semantic Query



A *semantic query* is developed here as means for precisely and **structurally** describing ontology requirements.

Element Similarity Score

represents the similarity of two ontology elements *x* and *y*.

For $x,y \in C$ or $x,y \in P$ or $x,y \in D$:

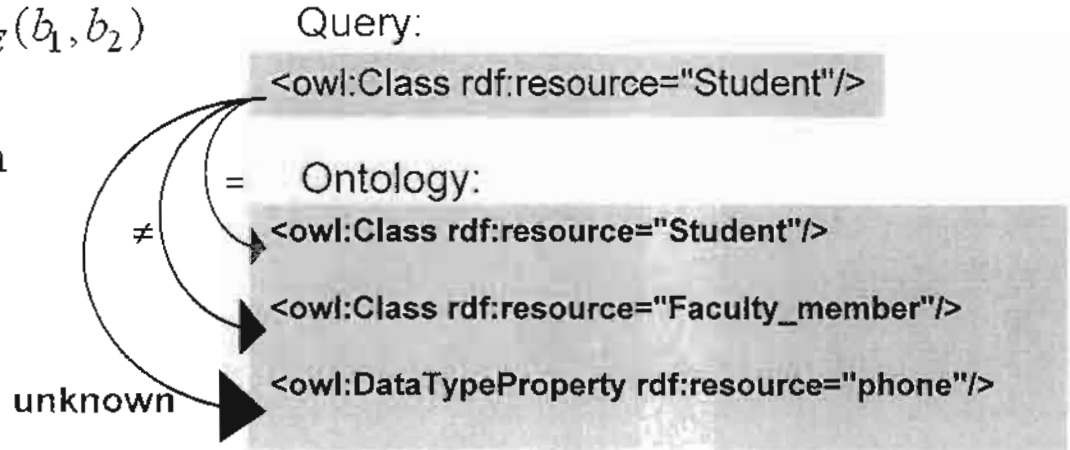
$$SS_E(x,y) = \begin{cases} w_{=} & : \text{if } x = y \\ w_{\equiv} & : \text{if } x \equiv y \\ w_{\supseteq} & : \text{if } x \supseteq y \\ w_{\subseteq} & : \text{if } x \subseteq y \\ w_{\neq} & : \text{otherwise} \end{cases}$$

For $x = m(a_1,b_1) \in R$ and $y = m(a_2,b_2) \in R$:

$$SS_E(x,y) = SS_E(a_1,a_2) * SS_E(b_1,b_2)$$

Otherwise:

$$SS_E(x,y) = \text{unknown}$$



Best Element Similarity Score

measures the similarity between an element $x \in C \cup P \cup R$ and an ontology $O \in ODB$

$$SS_B(x, O) = \max_{y \in M(O)} SS_E(x, y)$$

$SS_B(q,O) = w_=$

Query:

```
<owl:Class rdf:resource="Student"/>
```

Ontology O:

```
<owl:Class rdf:resource="Student"/>

<owl:Class rdf:resource="Faculty_member"/>

<owl:DataTypeProperty rdf:resource="phone"/>
```

Satisfaction Score of Conditions

Mandatory conditions:

$$SS_M(Q,O) = \begin{cases} 1 & : \text{if } mcond(Q) = \varnothing \\ 0 & : \text{if } \exists (c \in mcond(Q))\, SS_B(c, O) \leq w_{\neq} \\ \dfrac{\sum_{c \in mcond(Q)} SS_B(c,O)}{|\, mcond(Q)\,|} & : \text{otherwise} \end{cases}$$

Optional conditions:

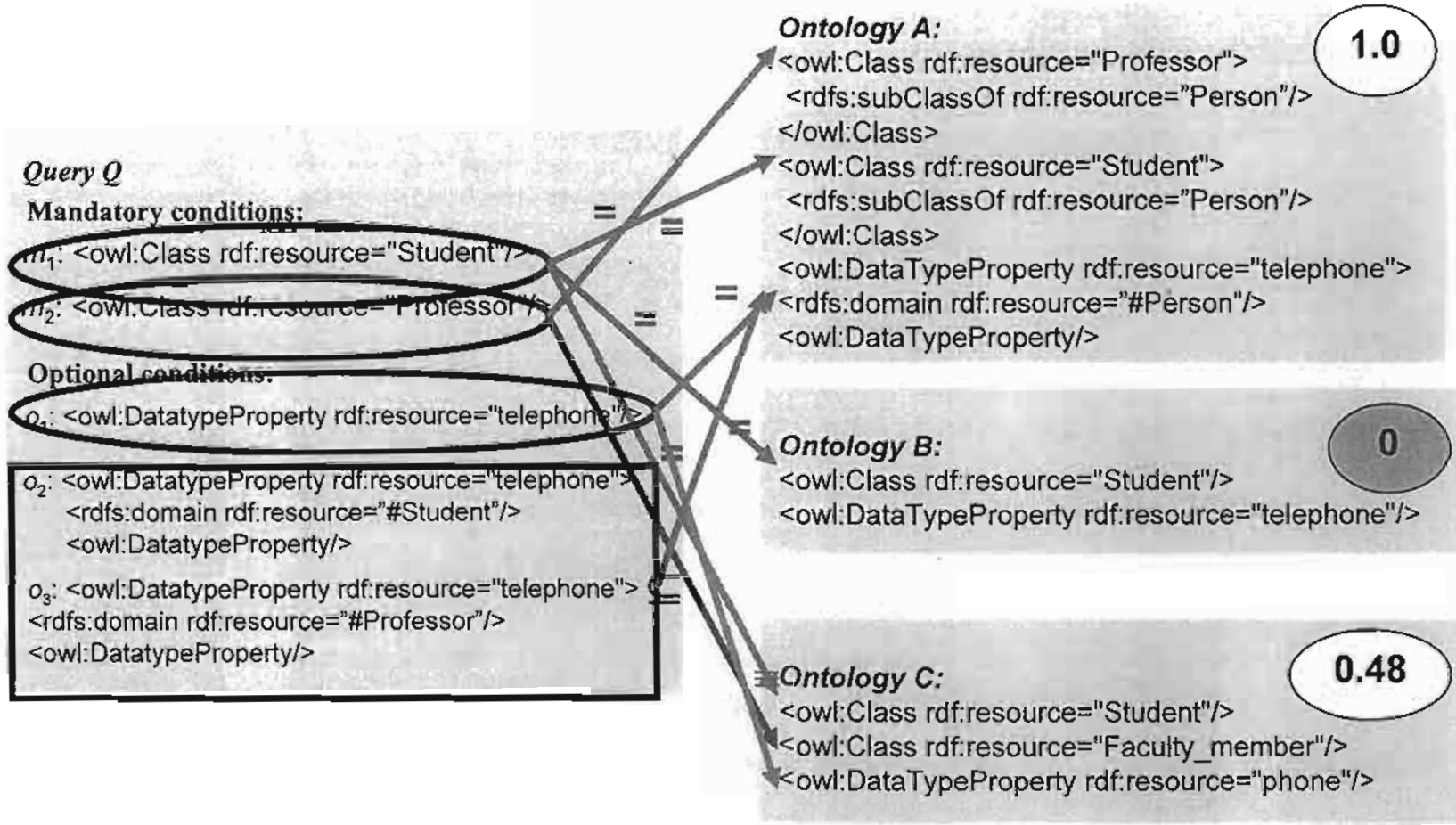
$$SS_O(Q,O) = \begin{cases} 1 & : \text{if } ocond(Q) = \varnothing \\ 0 & : \text{if } SS_M(Q,O) = 0 \\ \dfrac{\sum_{c \in ocond(Q)} SS_B(c,O)}{|\, ocond(Q)\,|} & : \text{otherwise} \end{cases}$$

Query-Ontology Similarity Score

represents the similarity between a semantic query Q and an ontology O

$$SS(Q,O) = w_M SS_M(mcond_Q, O) + (1 - w_M) SS_O(ocond_Q, O)$$

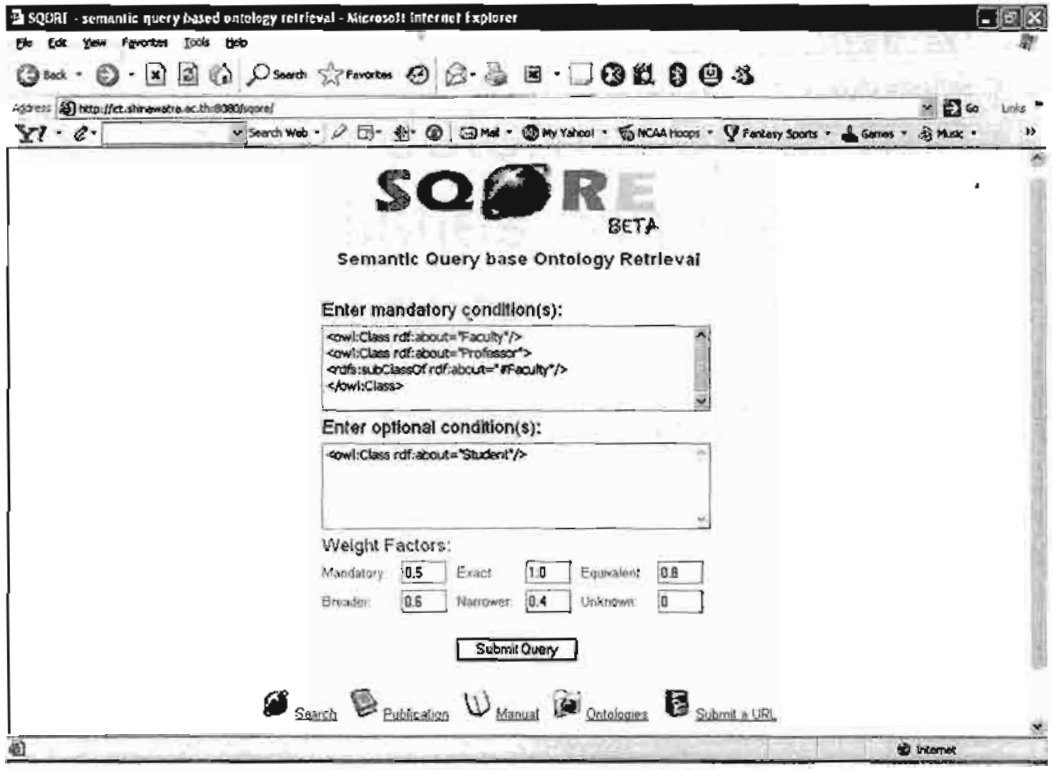
An Example



$W_M = 0.5$ $W_{=}=1.0$ $W_{\equiv}=0.8$ $W_{\supseteq}=0.6$ $W_{\subseteq}=0.4$ $W_{\neq}=0$

SQORE Prototype System

http://ict.shinawatra.ac.th:8080/sqore



Future Work

- *Compare* SQORE with other Ontology Retrieval Techniques
- *Extend* SQORE to determine the minimum set of ontologies that satisfies complex multidisciplinary application requirements
- *Deploy* SQORE to support exploratory search for non-expert users

Q&A