Role Based Access Control for the interaction with Search Engines

**Alessandro Bozzon[♮], Tereza Iofciu[◇], Wolfgang Nejdl[◇], Antonio Vincenzo Taddeo[□], and Sascha Tönnies[◇]

{bozzon}@elet.polimi.it, {iofciu,nejdl,toennies}@l3s.de, {taddeo}@alari.ch
[♮]Politecnico di Milano , P.zza L. da Vinci 32, I-20133 Milano, Italy
[◇]Forschungszentrum L3S, Appelstr. 9a, 30167 Hannover, Germany
[□]ALaRI, Faculty of Informatics, University of Lugano, Lugano, Switzerland

Abstract. Search engine-based features are a basic interaction mean for users to find information inside a Web-based Learning Management Systems (LMS); nonetheless, traditional solutions lack in mechanisms for access rights management for data contained in search engines' indexes. This paper explores the integration of a Role Based Access Control (RBAC) mechanism for the interaction with a search engine in a Web-based LMS. We first outline a reference conceptual model for the design of Web-based LMSs exploiting RBAC by means of WebML, a visual modeling language for the high-level specification of data-intensive Web applications. Then, we propose a model-driven approach for the definition of a RBAC-driven interaction between users and search engines, extending WebML with new modeling primitives and outlining significative modeling patterns for the specification of the *visibility* and *action* access control levels.

Key words: Web Engineering, Search Engine Design, Index Modeling, Access Control Modeling

1 Introduction and Motivation

Web-based Learning Management Systems (LMS) are gaining large consensus in several organizations. They are especially developed in the world of e-learning, as well in as general-purpose Web authoring tools and video-conferencing products. Being large-scale and multi-user applications, Web-based LMS are developed to manage a huge volume of data; their advantages can be recognized in their easy to use user-interface and in the common and customizable technological background provided by a Web application. Such extended collection of information is basically composed of documents, constituting the know-how on which the learning experience relies. These considerations suggest a clear need for an efficient knowledge sharing and knowledge management system. Therefore, a key point for an e-learning system is to implement a Knowledge Repository (KR) with access rights mechanism for the management of its resources: the simplest way to do this is by assuming the presence of a centralized repository, containing information entities for *documents*. This structure can be enhanced, for instance,

** In alphabetical order

by adding other information entities to categorize documents, to store authorship information, to label a set of documents and so on. Such model implies that all the documents are stored without any specific document access control. Nevertheless, in a multi-user web application not all the documents have to be available to all the users. In this situation, an access control mechanism is required in order to grant permissions for document managing only to authorized persons. This scenario applies also for the search features offered by LMSs: when searching for information in the KR, users should be allowed to consult only the resources for which they have proper permissions by filtering the overall collection of results for the performed queries against their assigned access rights. In data-intensive Web applications traditional searching functionalities are based on site navigation and database-driven search interfaces with exact query matching and no results ranking: such approach easily complies with the need for an access control mechanism as the results filtering can be directly performed by the database when queried (if access information are stored and related with the repository data); nonetheless, this approach is less effective w.r.t search engines (using information retrieval, IR, techniques), which, on the contrary, provide keyword-based queries, ranked results, and better performances in managing a lot of unstructured textual data residing outside the database. On the other hand, integrating IR functionalities within the Web applications introduces problems of data integrity, portability and run-time performance: implementing an access control mechanisms results in an additional effort, which can be solved only by means of integration software designed for the specific adopted technologies, databases schema and applications.

This paper explores the integration of an access control mechanism for the interaction with a search engine in the frame of the Web Modeling Language (WebML), a visual modeling language for the high-level specification of data-intensive Web applications and the automatic generation of their implementation code. The aim is to overcome the aforementioned problems by using a model-driven approach to declaratively specify (and automatically generate) search engines' index structures, content composition and access rules. We leverage on the role based access control (RBAC) method, which exploit the association between *users* and *permissions* through the assignment of user's *roles*: users can be assigned to roles, having associated permissions and, thus, users acquire permissions by having roles. We propose a high flexible RBAC for being applied to any kind of indexed resources either documents, or users, project and so on.

1.1 Driving Scenario

In order to exemplify the integration of a RBAC system in the management of a search engine, we refer to a simple e-learning scenario, in which a generic organization makes its knowledge repository (consisting in a collection of resources) available for consultation and, eventually, modification, to registered users. The

application in question is specified and developed through WebML [1] and incorporates a search engine for the retrieval of data stored both as database tuples and external documents (e.g. in TXT, PDF or DOC formats). The application exploits a RBCA system for the hypertext-driven management of (i) raw information stored in the database and for (ii) indexed information stored into a search engine's indexes.

2 Enabling Methodologies

2.1 Role Based Access Control Model with WebML

In this section we introduce a reference model for the design of a Web application exploiting RBAC. Although the approach we propose relies on the WebML language [1], we stress its adaptivity to most Web engineering methodologies.

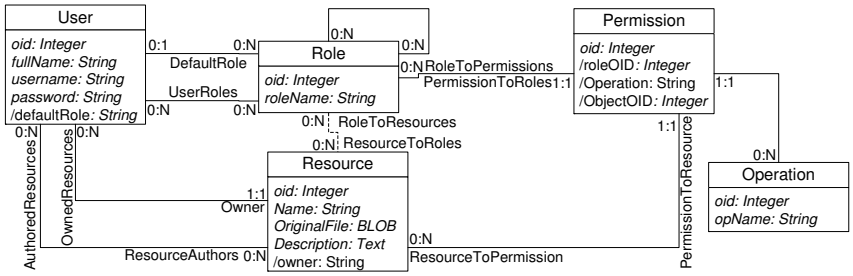

Fig. 1. Access Rights data model. Dashed lines and attribute with Italic font represents derived information.

Data model for RBAC In WebML, the content to be published is modeled using Entity-Relationship (E-R) or UML class diagrams. Figure 1 depicts the data model for a RBAC system, applied to our driving scenario. The reader can recognize the classical entities characterizing the reference model[2]: the *user* entity represents the Web application users; a *Role* is defined as a job function within an organization. Each role has associated semantics regarding the authority and responsibility conferred to the set of users belonging to it: a role can be assigned to different users, and a user can have different roles. A `Resource` is an information container (stored in the the knowledge repository) whos access has to be protected. `Permission` is an operation authorization given on protected resource. `Operations` are system dependent, and they refer to a set of executable functions for the user; for instance, in our case the allowed operations related to a resource can be the classical CRUD operations. In the schema of Figure 1, the entity `Permission` represents a many-to-many ternary relationship among `Role`,

[1] This scenario is suggested by the COOPER Initiative; see http://www.http://cooper-project.org. A running example of a RBAC system with WebML has been developed by ALARI

`Operation` and `Resource`. Thus, each role has associated a permission which allows an operation to a protected resource. This design of the data schema provides high flexibility and granularity for managing permissions to roles and users to roles. An important aspects of a RBAC is the concept of *role hierarchies* (RH). We managed such a key feature introducing a self-relationship on the `Role` entity with many-to-many cardinality in both directions (see Figure 1). Role hierarchies reflect the natural organization of roles in structured levels of responsibility and authority. Permission inheritance among roles has to be considered in a role hierarchy. In the case of web applications, we can add some constraints on the number of roles, the type of role hierarchies, and the allowed operations. Eventually, these constraints can be also managed through an ad-hoc web interface. Finally, an ownership relationship, between `User` and `Resource`, has been added. Usually, the owner of an object has some specific grants on the object, just for being its creator. These particular privileges are usually applied when the objects are documents, like in our case. These features have been captured from the current approach adopted in operating systems to manage file permissions (e.g. unix based).

Hypertext model for RBAC In WebML the hypertext model specifies the organization of the front-end interface of a Web Application by providing a set of modeling primitives for the specification of its publication (*pages* and *content units*), operation (*content units*) and navigation (*links*) aspects [1].

In a web application, the RBAC is used to control the user's access rights on the KR. Two levels of access control are considered for each user logged into the web application: the *visibility* level and the *action* level. In the former level, *visibility*, we filter the KR obtaining all documents visible to the logged user; the WebML model depicted in Figure 2(a) shows how, first, the user's role(s) having a *read* permission are retrieved and, then, documents are queried against such role(s), with an additional check over their ownership relationship.

In the latter level, *action*, once a visible document is selected by an user, its permissions are considered in order to provide the user with the list of allowed operations. Figure 2(b) shows how to filter, using WebML units, over such a list. For example, depending on the presence of the *modifiable* permission associated to the current user, the *Modifiable* data unit is shown, showing a link which allows the user to get to the modification page for the current document. *Visibility* and *action* can be also addressed in the design of the hypertext model by limiting the access to protected pages; in fact, WebML allows the definition of different *site views*, targeted to different user roles.

2.2 Modeling Search Engines' Indexes and Interaction

[3] introduced a conceptual model for data-intensive Web applications supporting search engines integration. An additional design layer (the *index data model*) enables (i)the specification of the search engine' indexes and their structure and (ii)the mapping between the resources of the Web application and their indexed representation in the search engine.

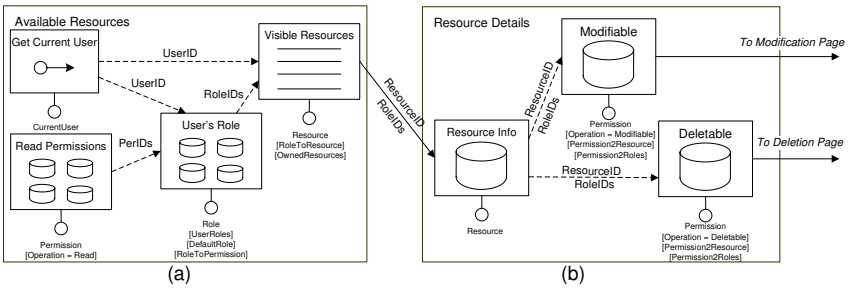

Fig. 2. WebML model of an example page to (a) get all the documents visible by a logged user and (b) manage the different document's operations based on the role of the logged user.

In the index data model, an *index* represents a description of the common features of a set of resources to index and, hence, contains a collection of *documents* having uniform structure. An index *document*, the indexed representation of a resource, is composed of a set of *fields* representing the properties of the index data that are relevant for the application's purposes. In order to uniquely distinguish an index document, a field named *object identifier*, or OID, is defined. Additional fields are *typed* by assigning them to a particular domain (e.i. *String*, *BLOB* etc).

The mapping between the search engine and the indexed data source(s) is per-

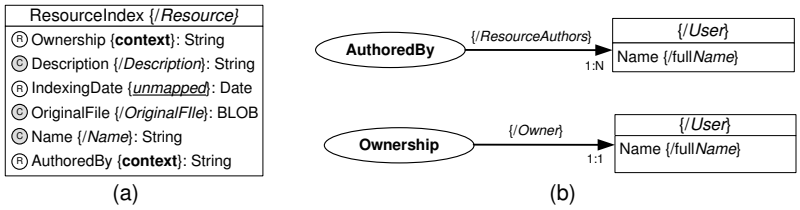

Fig. 3. Example of index data model with mapping information.

formed by specifying an association between an *index* in the index schema and an *entity* in the data schema. To uniquely identify the indexed information, the index will contain an additional implicit field, (named *entityOID* or EOID) containing the unique identifier of the indexed database tuple. Other *fields* are classified according to two orthogonal mapping dimensions: the *storing policy* specifies if the field should contain an indexed representation (*reference*) or a copy *cached* of the original data; in accordance with their *association*, instead, fields may contains data extracted from an entity's attribute (*Mapped* fields), data not directly derived from the database nor from indexed documents (*Unmapped* fields) or data originating from the context of the indexed entity (*Context*

fields) [2]. In the latter case, the content is composed aggregating data indexed from attributes belonging to entities related to the one defining the index; the aggregation expression is defined using the WebML data derivation language.

Figure 3(a) depicts an example of index data model for our reference application as well as its mapping over the data model. We modeled an index for the resources managed by the application (*ResourceIndex*) by specifying a set of fields composed by a subset of the associated data model entity's attributes (*Name*, *Description* and *OriginalFile*) plus further fields specifically aimed to enrich the information contained in the index in order to improve its retrieval performances (e.g. *AuthoredBy*). Figure 3 also reports the graphical representation of the data model mapping for the *ResourceIndex* index: for instance, the *OriginalFile* field is defined as *cached* over the *OriginalFile* attributes of the indexed entity while the *AuthoredBy* context field is defined as *reference* and its derivation expression representation is depicted in Figure 3(b). In order to

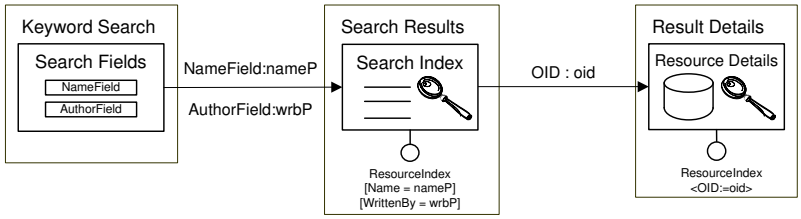

Fig. 4. Example of user to search engine interaction: result selection.

allow the specification of the interaction between the Web application, the user and the search engine, [3] also introduced extensions to the default WebML hypertextual primitives to cover the spectrum of operation specifiable for a search engine; a set of *content* units (the *Search*, *Document* and *Search Scroller* units) model the *publication*, in the hypertext, of documents extracted from an index defined in the index schema. For these units, the *source index* specifies the index to query while a *selector*, which is a predicate used to define queries over the search engine, specifies a (possibly empty) set of matching conditions over the *source* index's fields. Figure 4 depicts an example of usage for the *Search* and *Document* unit in our scenario: from the *Keyword Search* page, the parameters provided on the *Search Fields*'s outgoing links are used by the *Search Index* unit to perform a query on the index and to display its results. The selection of a result moves the navigation to the *Result Details* page, where the *Resource Detail* unit shows the details for the retrieved result. On the other hand, a set of *operation* units model the integration between the search engine and the underlying data back-end of a Web application in order to keep the content of the latter synchronized with the content of the former: the *Indexer* unit models the process of adding new content into an index to make it available for searching,

[2] By *context* we mean information semantically associated to a single concept scattered across different related entities in the data schema design process.

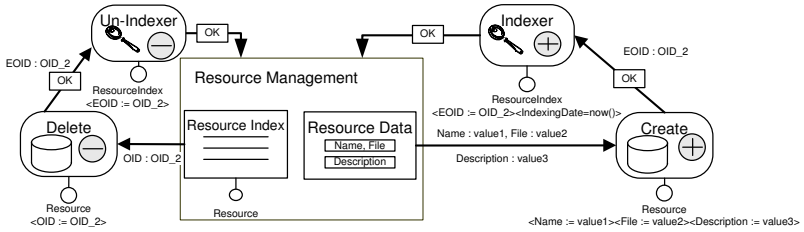

Fig. 5. Example of index management: index document creation and deletion.

the *Un-Indexer* unit models the process of removing content from an index and making it unavailable for search and the *Re-Indexer* unit models the process of updating the content of index documents by re-indexing again their source data. Referring to our scenario, Figure 5 depicts a typical example of usage for the *Indexer* unit (adding new resources into the knowledge repository and indexing it) and for the *Un-Indexer* unit (synchronization of the index after the deletion of a resource from the knowledge repository).

3 RBAC for interaction with Search Engines

Our model-driven approach aims at offering an homogeneous environment for the conceptual modeling and automatic code generation of Web-based LMS leveraging on fine-grained access policies to the content stored in their KR. Such approach requires the extension of WebML in order to apply the method illustrated in Section 2.1 to the modeling primitives described in Section 2.2; the integration is loose, in order to preserve the distinctive features of the two systems while allowing their independent evolutions and maintenance.

3.1 Defining permissions on the search engine

In RBAC systems, the list of grantable permissions for a user role includes the possibility of executing CRUD operations (*create*/*read*/*update* and *delete*) over the controlled resources. To apply such permission list over a search engine, we first have to define what it means for a resource to be *searchable* by a user: starting from the assumption that a user belonging to a role providing a *read* permission over a resource should be also allowed to search it, we define as *searchable* a resource whose content is accessible by a user. Transitively, we also define as *searchable*, by the same user, the indexed representation (in the search engine) of such resource. The same assumption, though, cannot be applied to the remaining CRUD operations: the *Create* operations represents a resource *indexing*, the *Update* operation represent a resource *re-indexing* and, finally, the *Delete* operation represent the removal of documents from the index. Such operations require a consistent computational effort, so this processes are usually performed asynchronously w.r.t. the KR management, typically by means of batch operation performed periodically by the search engine administrator(s).

We therefore can identify two different levels for the management (and, hence, the modeling) of permissions over the content of a search engine index: the (i)*document* level, and the (ii) *index* level. The *(i) Document level* tackles the problem of modeling the permission, for a user, to access the indexed content of a resource when performing a search. As stated above, such permission directly derives from the *read* authorization the user has on the original resource. To address this issue, we extended the *Index data model* with an additional *reference*, *unmapped* field, named *AccessRoles*: such field will contain the name of the user's roles having read access on the indexed document. Being part of the index, the *AccessRoles* field can be part of a query addressed to the search engine, allowing a fast and efficient filtering of the results accordingly to the given role(s). Figure 6(a) depicts the index model of Figure 3 modified with the additional field.
The *(ii) Index level*, instead, models the assignment of permissions to users

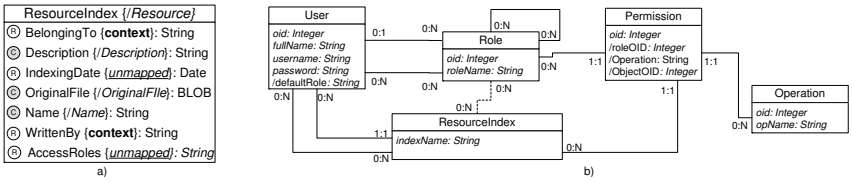

Fig. 6. Extended *Access Rights* and *Index Data* Model for RBAC

for createing/deleting and updating entries over an index. This is obtained by considering, in the access right data model, an *index* as an additional object managed by the RBAC mechanism. Indexes, hence, are simply represented as database entities (named *ResourceIndex*) defined by a unique identifier having the same name as the involved index. Figure 6(b) depicts a chunk of the access right data model of Figure 1 containing the additional entity: this simple yet effective modeling solution guarantees a uniform and centralized management of the access roles, while allowing an easy extension of the permission list to other operation specifically tailored for search engines.

3.2 Hypertext Model Extension

As introduced in Section 2.2, *searching*, *indexing*, *re-indexing* and *un-indexing* operations are represented in WebML respectively by the *Search*, *Indexer*, *Re-Indexer* and *Un-Indexer* units. In order to keep a loose integration between the RBAC method and the search engine, we adapted the hypertext models presented in Section 2.1 to the specific need of a search engine management: we still provide the *visibility* and *action* levels for access control by exploiting the existing content and operation unit, with the help of the existing RBAC management pattern shown, for instance, in Figure 2.

In details, *visibility* is addressed for the index content units by making use of the additional *AccessRoles* field introduced in Section 3.1: when defining a *Search Unit*, as in Figure Figure 7(a), an additional query predicate over such

field automatically expands the query submitted to the search engine with the list of roles currently executing the search; the retrieved results will be automatically filtered over such roles, with no further computational effort for the Web application. For the *action* level, instead, we use a slightly modified version of

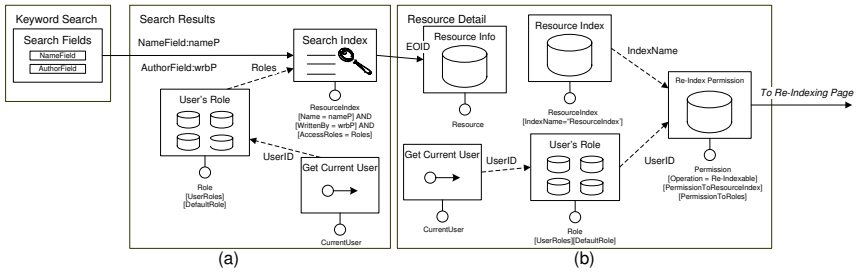

Fig. 7. Managing access rights for search engines in the hypertext model.

the pattern depicted in Figure 2(b): in Figure 7(b), for a selected resource, only allowed users will have the possibility of executing a re-indexing operation. On the other hand, when indexing a resource, also information about the roles having *read* permission have to be added, enabling access control of search results.

4 Related Work

In the last years, systems for collaborative work like the COOPER Platform [4] got increased popularity. For this kind of systems there is a strong need for effective search inside their knowledge sharing repositories, and the increasing amount of information in those context pushed forward further development of search infrastructures for enterprise data management systems [5]. However, the partial private nature of shared information makes the application of traditional document indexing schemes difficult: the problem of allying search-engines' ranking algorithms to access-controlled collections is outlined in [6]. The access levels and access control policies have to be reflected in the index structure and/or retrieval algorithms as well as in ranking the search results. In literature, several solutions address the problem defining privacy policies for data stored on remote servers, which typically provide the basis for the collaborative platforms. For example cryptographic techniques can enable users to store encrypted files on a remote server and retrieve them using keyword search [7][8]. But these solutions are not suitable for a collaborative multi-user environment making use of external search-indexes, resulting in a lack of methodologies for the definition of secure searching inside collaborative systems. Our solution aims at overcoming such lack by means of a role based access control model. Details on RBAC can be found in [2] and, as studied in [9], RBAC has shown its advantages in allowing security for web-based applications by performing large-scale authorization management.

5 Conclusions and Future Work

In this paper we have addressed the problem of the integration of an access control mechanism for the interaction with a search engine in the context of Web-based Learning Management Systems. We have proposed an extension for a specific modeling notation (WebML) to support the specification of search engine indexes enriched with access control information; in addition, we also proposed WebML reference patterns to model the access-controlled management of the typical operation specifiable for a search engine.

For future work, we plan to enrich the proposed access control solution with finer-grained policies working at field level, while providing new modeling primitives for a better specification of access policies management. We also intend to integrate the proposed extension to WebRatio[10] (the development suite based on WebML) to support the automatic generation of Web application integrating access rule control over the interaction with search engines.

References

1. S. Ceri, P. Fraternali, and A. Bongio. Web modeling language (webml): a modeling language for designing web sites. In *WWW9 Conference, Amsterdam, Holland.*, 2000.
2. David F. Ferraiolo and Ravi Sandhu and Serban Gavrila and D. Richard Khun and Ramaswamy Chandamouli. Proposed NIST Standard for Role-Based Access Control. In *ACM Transactions on Information and System Security*, volume 4, pages 224–274. ACM, August 2001.
3. A. Bozzon, T. Iofciu, W. Nejdl, and S Tönnies. Integrating databases, search engines and web applications: a model-driven approach. In *ICWE2007, Como, Italy*, 2007.
4. A. Bongio, Jan van Bruggen, S. Ceri, M. Matera, A. Taddeo, X. Zhou, and et. al. COOPER: Towards A Collaborative Open Environment of Project-centred Learning. In *EC-TEL'06, Crete, Greece - October 1-4, 2006*, 2006.
5. D. Hawking. Challenges in enterprise search. *Proceedings Fifteenth Australasian Database Conference*, 2004.
6. S. Buttcher and C.L.A. Clarke. A Security Model for Full-Text File System Search in Multi-User Environments. *Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST 2005).*
7. H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 216–227, 2002.
8. D.X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55, 2000.
9. J. B. D. Joshi, W. G. Aref, A. Ghafoor, and E.H. Spafford. Security models for web-baed applications. *Communication ACM*, 2:38–44, 2001.
10. Web Models s.r.l.