Using Ontology-Based Methods for Implementing Role-Based Access Control in Cooperative Systems

Satyajeet Raje Chowdary Davuluri Michael Freitas Rajiv Ramnath Jay Ramanathan
Department of Computer Science and Engineering,
The Ohio State University Columbus, Ohio, USA.
ramnath@cse.ohio-state.edu

ABSTRACT
Cooperative systems and Internet-based collaborative environments nowadays are becoming pervasive. The issue of security of data becomes very critical due to the federated databases that such systems integrate. In this paper, we describe the implementation and evaluation of a role-based access control (RBAC) mechanism for a system used to support proteomics researchers in collaborative project group at a major medical center at a R1 research university. This system uses ontology-based methods for its implementation. Using an ontology in RBAC has several advantages. It eases the process of making modifications. It also brings about standardization, which is cornerstone for portability. We test and evaluate this approach in an implementation of a data-management system for proteomic experiment data. The primary aim of this study is, firstly, to make use of an upcoming and potentially standard technology and apply it to the domain of system security. Our second aim is to validate the hypothesis that such a method can be effectively used in a real-world cooperative system.

Categories and Subject Descriptors
H.2.7 Database Administration

H.5.3 Group and Organization Interfaces

C.1.3 Other Architecture Styles

General Terms
Management, Security and Standardization.

Keywords
Ontology, Access Control, RBAC, Semantic Web.

1. INTRODUCTION
The safety and consistency of information are not trivial issues for even the smallest of organizations. In collaborative research environments in particular, addressing data access-control issues is very important, but difficult to find solutions to. The scale of these issues becomes even more severe when databases serving these collaborative environments are federated and heterogeneous i.e. are on different platforms and varied in their the schema. It has also is become crucial to have standardized mechanisms for access control. The reason is that this facilitates collaboration and allows for faster cooperation. Standardized mechanisms provide the ability to manage large cooperative systems.

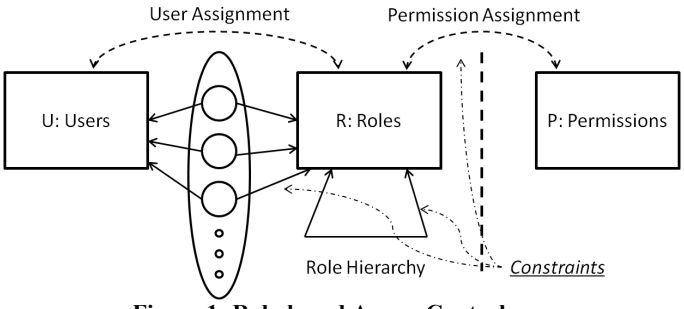

Figure 1: Role-based Access Control

Role-Based Access Control (RBAC) and Team-based Access Control (TMAC), an extension of RBAC, are techniques considered suitable for managing access in cooperative organizations [1, 2]. RBAC, explained in Figure 1 above, was introduced in early multi-user computer systems [3, 4]. As seen in the figure, RBAC separates the user management and assignment of permission. A major advantage of RBAC is its ability to constraint access based on the concept of separation of duties, which significantly simplifies the management of permissions, because it is easy to use and understand. RBAC is a means for controlling access to resources based on the roles that individual users have within an organization. In this method, individual users are assigned roles, which, in turn, are associated with permissions, In other words, instead of specifying access rights (read, write, etc.) to individual objects a user is granted access based on his assigned roles. RBAC has stirred up interest from the research community working in data and information security as can be seen in [2]. However, the success of traditional RBAC techniques comes at a price. Because of the additional level of indirection in the specification of the access-control policy, these techniques lack the granularity that is required for effective data access control as the permissions are restricted to the roles and making exceptions is not easy. In addition, traditional RBAC also cannot utilize contextual information as is required in larger collaborations. This makes the traditional RBAC cumbersome and not as effective as expected in the cooperative environment.

The method described in this paper tries to overcome these shortcomings by using an ontology-based approach for specification and implementation of the RBAC in a collaborative system used within a research group to manage proteomics data, where the access control policy depends on how the project team hierarchy is structured.

2. USING ONTOLOGY BASED MECHANISMS FOR RBAC
Traditional RBAC techniques are typically difficult to adapt across organizations [5]. If a good access control mechanism has been implemented for one project, it is difficult to modify and use

the same for another project. This is a major concern in case of dynamic project environments, where it is difficult for traditional RBAC to cope with changing organizational structures, user roles and security requirements found in such scenarios. As dynamic environments are becoming common there is need of change in the RBAC techniques as well.

Over the past few years, ontology-based methods are being used to solve problems with a wide range of scope [6]. In this study we apply these methods to address access control in organizational structures. They permit the uniform description of organizational structures, roles, privileges and resources at different levels of abstraction and supports reasoning about both the structure and the properties of the elements that constitute the system. Essentially, ontology-based mechanisms allow for contextual information to be stored along with the access control mechanisms. Another major advantage is that ontology-based systems require only the URIs to point to the actual data sources URI and URI handling is an integral part of Web 3.0, so moving to a federated system is easy and an implementation based on ontological methods provides this inherently. This becomes very important in collaborative environments as the data in question is invariably distributed. However, since the control to the data is entirely URI based it does not matter where the data is located as long as we have an updated URI. We feel that all these advantages make up for the shortfalls in traditional RBAC and make it more suitable for cooperative systems with heterogeneous environments. Finally, ontology-based mechanisms provide the standardization essential for portability [7].

The researchers in [1] provide a comparative analysis of the different possible access control mechanisms. They conclude that that decentralized access control models that use Web 3.0 technologies show promise for federated, collaborative systems. A similar comparison is provided by [2]. In [8] ontological methods for access control to web communities is discussed. There are two types of RBAC constraints: dynamic and static. Authors in [9] described an approach for RBAC with dynamic constraints using automated reasoning techniques.

In [10], authors presented an approach to reduce the inefficiencies of the management (coordination, verification and validation, and enforcement) of many role-based access control policies and mechanisms using OWL[1]. They focused on the representation of XACML[2] (eXtensible Access Control Markup Language) policies in DL. The authors in [11] discuss how one might use SPARQL along with the reasoner to implement RBAC on the semantic web. In [12], the authors also suggested expressing access control policies based on OWL and SWRL. The solution was limited to the definition of OWL ontology and declaration of SWRL rules. They predicted the use of an engine to deduce more information by adding rules. Another such technique is discussed in [13]. The proposed solution actually uses an OWL reasoner called Pellet[3] to execute rules and deduce more information. Paper [14] also proposed using OWL for constructing ontologies that define policies/privileges.

[1] OWL Web Ontology Language Reference: http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[2] http://www.oasis-open.org/committees/xacml/

[3] Pellet: The Open Source OWL 2 Reasoner: http://clarkparsia.com/pellet/

Organizational semantics and access control mechanisms may be formally represented in an ontology using several available ontology languages. In our work, we focus on Web Ontology Language (OWL) recommended by the W3C. The system can control access to the resources of the organization by providing differential access privileges. We specify static constraints on roles, and use Semantic Web Rule Language (SWRL) rules over the ontology to infer new knowledge to be passed back to the ontology. Through these rules, verification of access control constraints defined in the ontology are also achieved. Our evaluation shows that the proposed solution can adapt to changing organizational structures with less effort, and (finally) that there is acceptance from users of the system.

3. AN EXAMPLE ORGANIZATIONAL STRUCTURE

For illustration let us look at the organizational structure in Figure 1 below. This example is based on an actual proteomics research group within a comprehensive cancer center (CCC) within the medical center at a major R1 university.

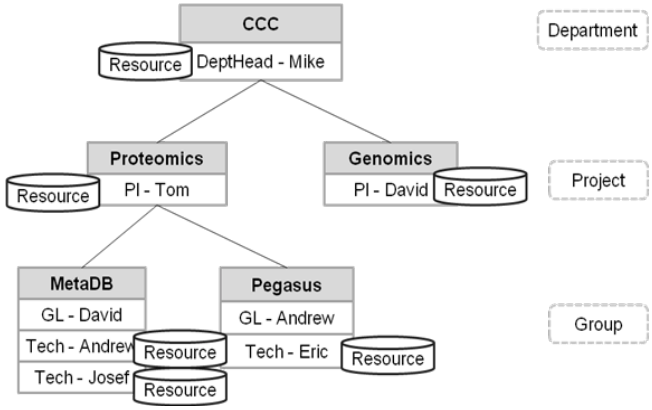

Figure 2: A typical organizational structure

This real example is not unlike a hierarchy within a dynamic collaborative project group. It deals with the roles Department Head (DeptHead), Principal Investigator (PI), Group Leader (GL) and Technician (Tech). CCC (a Department) contains two projects - Proteomics and Genomics. Proteomics (Project) in turn contains groups MetaDB and Pegasus. A department is headed by a Department Head, a project by a Principal Investigator and a group by a Group Leader, respectively. A group may contain multiple technicians. A researcher's resources consists of the set of files owned by him. A person may have multiple roles. For example, David is not only a PI (Project – Genomics) but also a GL (Group – MetaDB). He should have access to the corresponding project and group resources in which he is involved. Thus, a person's access rights are determined by his role in the departments, projects and groups to which he belongs. Following are some of the examples of access rights for the specific roles:

1. DeptHead is the head of a Department. He can read the files owned by PIs of all the projects owned by his department. He can add and delete projects to his department. He can also add and remove PIs to the projects within his department.
2. PI is the head of a project. He can read the files of GLs of all the groups owned by his project. He can add or delete groups

to his project. He can also add or delete GLs to the groups his project owns.

3. GL is the head of a group. He can read and write the files of all the Technicians of his group. He can add and delete Technicians to his group.
4. A Technician can read the files of all the others Technicians in his group.
5. Finally, Department Heads gain read permission to files of Principal Investigators of projects part of their respective departments.

Note that the system should manage access to resources not only based on roles but also based on the involvement in organizational divisions (departments, projects and groups). The traditional RBAC implementations cannot provide support for such specific role assignments.

4. THE PROPOSED ONTOLOGY

4.1 Defining Concepts

The ontology described here is based on the specific organizational structure described in Figure 2. In the proposed ontology, the *Researcher* class defines researchers of the organization. We specify *Department*, *Project* and *Group* as subclasses of *WorkUnit* in order to avoid defining explicit relationships between department / project / group and roles. Role is the positional hierarchy of researchers in the organization. Through this, individuals are restricted in access to the correct resources. The class hierarchy is a critical issue in inheritance of properties. The classes *Department Head*, *Principal Investigator*, *Group Leader* and *Technician* are a subset of the *Role* class. The *File* class defines the files owned by a researcher. The *Tag* class defines user-defined folksonomies.

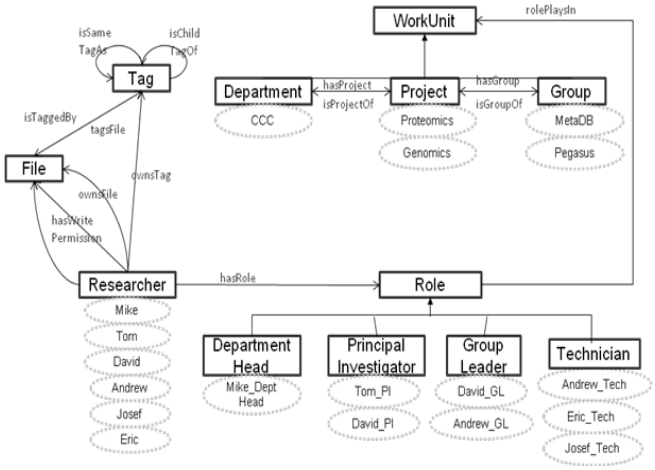

Figure 3: The proposed ontology

4.2 Concepts through Instances

OWL classes are interpreted as sets that contain instances of concepts (i.e. individuals). In Figure 3 instances of classes are shown in ellipses. Instances of the researchers are defined here simply as their names. Instances to the subclasses of *Role* are added in accordance with the individual roles to realize multiple roles of a person. As for example, *David_PI* instance of *Principal Investigator* corresponds to the principal investigator instance of David. Similarly, *David_GL* corresponds to the group leader role of David.

4.3 Relations through Properties

Properties are the binary relations between the two things, more specifically between the instances of classes. A property relates instances from the *domain* with the instances from the *range*. Syntactically, a *domain* links a property to a class and *range* links a property to either a class or a data range. Due to the class hierarchy and *domain* and *range* specifications, subclasses inherit the relationships between the respective classes. Figure 3 also provides the properties and their relationships with classes. *rolePlaysIn* property specifies the fact that in which specific *Work Unit* (department/project/group) a *Role* instance plays its role. The proposed ontology is supposed to answer what kinds of permissions a researcher has on a resource and *hasReadPermission* and *hasWritePermission* properties define this situation. Note that the relationships of *hasReadPermission* and *hasWritePermission* are not defined explicitly. These relationships of the properties are filled in through the inferencing process.

5. ENHANCING THE KNOWLEDGE BASE

The expressivity provided by the OWL is limited to tree like structures [15]. This means that knowledge based on indirect relations between the entities cannot be inferred from an OWL ontology. Therefore, we do inferencing using rules described over the ontology, using the Semantic Web Rule Language (SWRL)[4] - a complementary feature of OWL. SWRL rules can be used to infer new knowledge from existing OWL knowledge bases. SWRL is based on a combination of the OWL DL and OWL Lite sublanguages of the OWL and the Unary/Binary Dialog sublanguages of the Rule Markup Language[5] (RuleML). SWRL allows users to write Horn-like rules expressed in terms of OWL concepts to reason about OWL individuals without creating restrictions to the original functionality [16]. In this work we also used the Pellet reasoner explained in [17]. Pellet is a complete OWL-DL reasoner with acceptable to very good performance, extensive middleware, and a number of unique features that we have used to implement the rules. It is the first sound and complete OWL-DL reasoner with extensive support for reasoning with individuals (including nominal support and conjunctive query), user-defined data types, and debugging support for ontologies. It implements several extensions to OWL-DL including a "combination formalism" for OWL-DL ontologies, a non-monotonic operator, and preliminary support for OWL/Rule hybrid reasoning.

Objects of the properties *hasReadPermission* and *hasWritePermission* are filled in through the inferred knowledge derived by executing the rules. In order to do this, the proposed ontology and SWRL rules are transferred to Pellet. Running the reasoner then initiates the inferencing process, which generates additional knowledge elements that are passed back to, and enrich, the ontology. Some of the SWRL rules corresponding to the example policies in Section 3 are as follows:

- **Rule1:** A group leader has write permission over the files owned by all the technicians in his group

[4] http://www.w3.org/Submission/SWRL/

[5] http://ruleml.org/

File(?f) , Group(?G) ,
Researcher(?p) , Researcher(?t) ,
GroupLeader(?p_GL) , Technician(?t_Technician) ,
hasRole(?p, ?p_GL) , hasRole(?t, ?t_Technician) ,
isFileOwnedBy(?f, ?t) ,
*rolePlaysIn(?p_GL, ?G) , rolePlaysIn(?t_Technician, ?G) -> **hasWritePermission(?p, ?f)***

- **Rule2:** A technician has read permission over the files owned by all the other technicians in his group

File(?f) , Group(?G) ,
Researcher(?p) , Researcher(?t) ,
Technician(?t_Technician) , Technician(?p_Technician) ,
hasRole(?p, ?p_Technician) , hasRole(?t, ?t_Technician) ,
isFileOwnedBy(?f, ?t) ,
*rolePlaysIn(?p_Technician, ?G) , rolePlaysIn(?t_Technician, ?G) -> **hasReadPermission(?p, ?f)***

- **Rule3:** Department Heads gain read permission to files of Principal Investigators of projects part of their respective departments.

Department(?D) , Project(?P) , File(?f) ,
isProjectOf(?P, ?D) ,
Researcher(?p) , Researcher(?t) ,
DepartmentHead(?p_DeptHead) , PrincipalInvestigator(?t_PI) ,
hasRole(?p, ?p_DeptHead) , hasRole(?t, ?t_PI) ,
isFileOwnedBy(?f, ?t) ,
rolePlaysIn(?p_DeptHead, ?D) , rolePlaysIn(?t_PI, ?P)
*-> **hasReadPermission(?p, ?f)***

Once again, note that all the relationships that exist between the entities in the ontology are not explicitly defined in the ontology. For example, *hasWritePermission* relationship of group leader David (*David_GL*) has not been explicitly defined. Executing Rule 1 infers those relationships, and identifies which resources *David_GL* has write access to. That is (see Figure 2) executing Rule 1 infers that researcher David, as group leader of MetaDB, has write permission to the resources owned by researchers Andrew and Josef. The inferred results are exported back to the ontology to fill these empty relationships. Execution of Rule 2 shows the similar result. Researcher Andrew, as a technician in the MetaDB group, gains read access to the resources owned by researcher Josef. The implementation of other rules is also straightforward.

6. IMPLEMENTATION

The research described in this paper is part of a larger effort to provide a usable, extensible, scalable system and framework (called MetaDB) for management of large-scale proteomics data for research groups within a large medical research center. Figure 3 shows a component diagram for MetaDB. The MetaDB framework provides APIs for managing meta-data, fetching consolidated logical data sets, updating information objects etc. This component is domain independent, very generic and can be used to manage almost any kind of data (for example, a shared music collection, something like a federated iTunes™!). The core component interacts with different data sources using standard access methods – SOAP-based web services and a ReSTful interface. The details of the implementation are provided in [18].

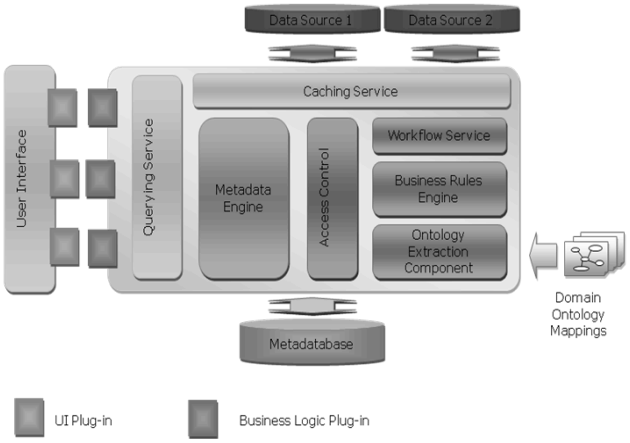

Figure 4: Component Diagram

The framework (including the core component) is built on the Enterprise Java framework. It is extensible in that it provides for adding on domain specific functionality through plug-ins. For example, there is a proteomics plug-in to handle proteomics metadata. Domain specific plug-ins must also provide a corresponding UI component. The primary purpose of the plug-ins is to create a specific view of data.

An example of the access-management GUI is shown in Figure 4, which shows the panel from which the administrator views and modifies the organizational hierarchy, by adding or removing roles, rules and individuals.

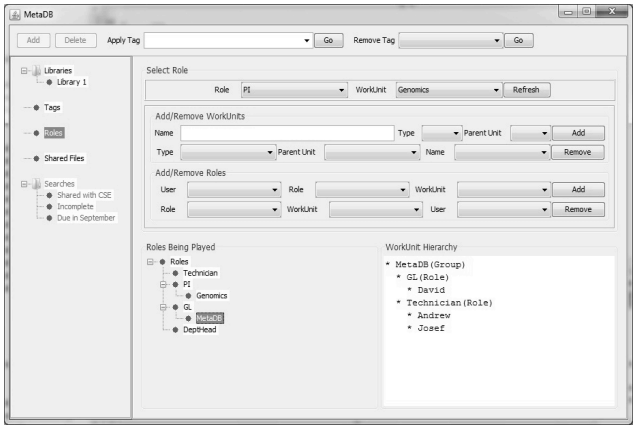

Figure 5: GUI for the tool

7. EVALUATION

In this section we present a brief evaluation of the tool along the axes of flexibility, usability and performance.

7.1 Flexibility

The ontology-based approach allowed us to reflect the organization changes of a research environment with minimum effort. For example, let us assume that a new group, MicroArrays has been created in the project hierarchy shown in Figure 1, and is to be added to the Genomics project, with researchers Adam and Josef having joined as the group leader of and technician within the group respectively. These changes in the organizational

structure may be straightforwardly accomplished using the user-interface shown in Figure 3. Now, let us see how the access-rights to files are adjusted.

It is expected that a group leader (Adam) has read permission over the files owned by the technicians (Josef) in his group. As a principal investigator, researcher David gains read and write permissions to the files owned by research Adam, group leader of group MicroArrays, under project Genomics. The corresponding actions to reflect these changes into the ontology are described below, as a means of evaluating the strength of our approach.

1. Add new researcher instance: *Adam*
2. Add new group instance: *MicroArray*
3. Add new instance of *Group Leader* role: *Adam_GL*
4. Calculate the corresponding relationships.
5. If we execute Rule 1 (described in Section 5), new relationships are inferred with *hasWritePermission* property. These are exported back to the ontology to fill in the empty relationships. As expected, researcher David gains read access to the files owned by researcher Adam.

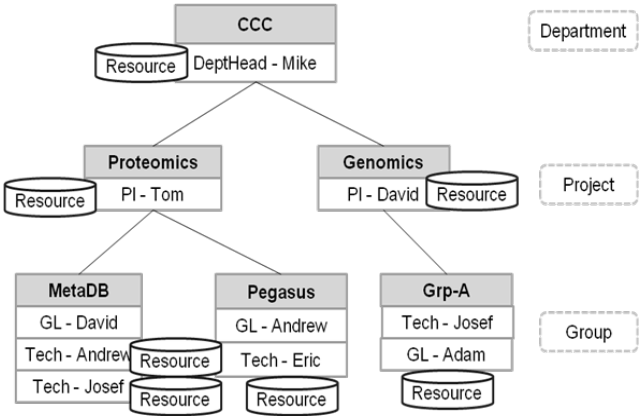

Figure 6: Modified Structure

Note that the changes that are required to the existing ontology are all in the instances, and automatically done by the rules. In our approach it is also simple mechanism to revoke somebody's role or privilege. One can simply delete the relationship between the role instance and the corresponding permission instance to withdraw the privilege. Afterward the corresponding role instance itself can be deleted to entirely repeal the subject's role-specific permissions.

7.2 Performance

Although OWL ontologies can now be stored in relational databases, querying resulting from the inferencing process was computationally very expensive, with frequent querying drastically reducing the performance of the system when managing large numbers of entities. Thus, MetaDB architecture essentially copies into the ontology the contents of the database. This replication of information resulted in significant run-time performance gains in read-only situations. However, when new entities (data objects, users and roles) were added the entire ontology was recomputed.

7.3 User Acceptance

A small user assessment of MetaDB has been conducted in order to gather requirements for future work. The users were developers not associated with this project (in order to minimize bias). The software implementation was provided as an executable jar file along with a basic Users' Guide. Users were asked to fill out a questionnaire in which they were asked to indicate their level of agreement with a set of assertions about MetaDB (with 1 indicating strong disagreement and 5 indicating strong agreement). Table 1 summarizes the results of the questionnaire.

	Statement	Average Rating
1	It helps me be more effective	3.25
2	It is useful	3.25
3	I can use it successfully every time	4.25
4	It saves me time when I use it	3
5	It meets my needs	3
6	It does everything I would expect it to do	4
7	It is user-friendly and simple to use	4
8	I can use it without written instructions	4.5
9	I don't notice any inconsistencies as I use it	3.5
10	I am satisfied with it	4

Table 1: User Evaluation

The evaluation showed that users evaluated the ease of use of the software positively, with almost all the users reporting that the user interface was intuitive and usable without written instructions. Interestingly, several of the users felt that the software would be more useful to their respective supervisors than to them. When we went deeper into this issue with them, users supported this assertion by saying that they knew well where their *individual* data was stored and how it was organized. When they were specifically asked about shared data, they began to better see the usefulness of the software. Also, it turned out that all the users participated in the study *were part of multiple projects, but with no clearly defined roles*. Thus, it was clear that they had not yet been placed in situations where access control was complex. Certainly, an expanded, comprehensive user evaluation is necessary.

8. DISCUSSION AND FUTURE WORK

Given the interpreted nature of the technology this approach is computationally expensive. Also, note that decidability is not guaranteed by SWRL. We are currently working on caching meta-date in the relational database for frequent and faster centralized access. We believe this will decrease the access time for the system drastically.

Another impact to performance was caused by the fact that the state of the art in Description Logic (DL) reasoners currently do not allow incremental reasoning (so that less re-computation is needed when updates to rules are made, or when new objects are added. (Although Pellet, the OWL-DL reasoner being used, contains support for incremental classification and incremental consistency checking, it still does not support incremental realization). Thus, changes to rules are currently implemented in a "clear-and-reload" manner. Integrating incremental reasoning within the current system, whenever available, should increase the performance, particularly when reasoning over large knowledge bases.

9. CONCLUSION

In this paper, we discussed how access management in dynamic project-based environments might be implemented using Semantic Web technologies. Specifically, we developed an

ontology to represent the organizational structure of a project-based dynamic and collaborative research environment and the roles of individuals. In the ontology, dynamic and non-hierarchical relationships between the entities could not be defined explicitly; semantic rules (in SWRL) were used to specify additional access control policies. A Pellet OWL-DL reasoner executed these rules to calculate new facts, which were then transferred back to the ontology. The system has been assessed at a small-scale and has received positive responses from potential users.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation's Industry-University Cooperative Research, at the Centre for Experimental Research in Computer Systems at Georgia Tech Institute and The Ohio State University, and by the National Institutes for Health Grant R01CA107106. We also thank research group members at the Ohio State Medical Center for their help in the evaluation. We also acknowledge Aman Kumar, who implemented the first prototype of metaDB.

REFERENCES

[1] Baina, A., Deswarte, Y., El Kalam, A. and Kaaniche, M. Access Control for Cooperative Systems: A Comparative Analysis. In *Proceedings of the 3rd Intl. Conf. on Risks and Security of Internet and Systems (CRiSIS '08)* (Tozeur, Tunisia, Oct 28-30, 2008), IEEE Computer Society, Piscataway, N.J., 2008, 19 - 26.

[2] Tolone, W., Ahn, G., Pai, T. and Hong, S. 2005. Access control in collaborative systems. *ACM Comput. Surv.* 37, 1 (March 2005), 29-41.

[3] Sandhu, R., Ferraiolo, D. F. and Kuhn, D. R. The NIST Model for Role Based Access Control: Toward a Unified Standard. In *Proceedings of the 5th ACM workshop on Role Based Access Control*, (Berlin, German, July 26-27, 2000). ACM Press, New York, NY, 2000, 47-63.

[4] Bacon J., Moody K., and Yao W. 2002. A model of OASIS role-based access control and its support for active security. *ACM Trans. Inf. Syst. Secur. 5, 4* (Nov 2002), 492-540.

[5] Elahi, N., Chowdhury, M. and Noll, J. Semantic Access Control in Web Based Communities. In *Proceedings of the 3rd Intl. multi-conference on Computing in the Global Information Technology (ICCGI '08)* (Athens, Greece, July 27-Aug 1, 2008). IEEE Computer Society, Los Alamitos, CA, 2008, 131 - 136.

[6] Hayes-Roth, F. and Jacobstein, N. 1994. The state of knowledge-based systems. *Commun. ACM* 37, 3 (March 1994), 26-39.

[7] Fensel, D. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, Berlin, NY, 2001. Print.

[8] Chowdhury M., Chamizo J., Noll J. and Miguel J. Capturing Semantics for Information Security and Privacy Assurance. In *Proceedings of the 5th Intl. conference on Ubiquitous Intelligence and Computing (UIC '08).* (Oslo, Norway, June 23-25, 2008). Springer-Verlag, Berlin, Heidelberg, 2008, 105-118.

[9] Dury A., Boroday S., Petrenko A. and Lotz V. 2007. Formal Verification of Business Workflows and Role Based Access Control Systems. In *Proceedings of the The International Conference on Emerging Security Information, Systems, and Technologies* (SECUREWARE '07). IEEE Computer Society, Washington, DC, USA, 201-210.

[10] Smith, M.A., Schain, A.J., Clark, K.G., Griffey, A. and Kolovski, V. Mother, May I? OWL-based Policy Management at NASA. In *Proceedings of the Workshop on OWL: Experiences and Directions (OWLED 2007).* (Innsbruck, Austria, June 6-7, 2007).

[11] Cirio, L., Cruz, I., and Tamassia, R. A Role and Attribute Based Access Control System using Semantic Web Technologies. In *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems - Volume Part II* (OTM'07), Robert Meersman, Zahir Tari, and Pilar Herrero (Eds.), Vol. Part II. Springer-Verlag, Berlin, Heidelberg, 1256-1266.

[12] Di W., Jian L., Yabo D., and Miaoliang Z. Using Semantic Web Technologies to Specify Constraints of RBAC. In *Proceedings of the 6th Intl. conference on Parallel and Distributed Computing Applications and Technologies (PDCAT '05).* (Dalian, China, December 5-8, 2005). IEEE Computer Society, Washington, DC, USA, 2005. 543-545.

[13] Li, H., Zhang, X., Wu, H., Yuzhong, Q., Design and Application of Rule Based Access Control Policies. In *Proceedings of the 4th Intl. Semantic Web and Policy Workshop (SWPS).* (Galway, Ireland, November 7, 2005)

[14] Finin T. and Joshi A. Agents, trust, and information access on the semantic web. *SIGMOD Rec.* 31, 4 (December 2002), 30-35.

[15] Motik B., Sattler U. and Studer R. Query Answering for OWL-DL with rules. *Web Semant.* 3, 1 (July 2005), 41-60.

[16] O'Conner, M., Knublauch, H., Tu, S., Grosof, B., Dean, M. Grosso, W., and Mussen, M. Supporting rule system interoperability on the Semantic Web with SWRL. In *Proceedings of the Fourth International Semantic Web Conference (ISWC2005).* (Galway, Ireland, November 6-10, 2005)

[17] Sirin, S., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y. Pellet: A practical OWLDL reasoner. Technical Report CS 4766, University of Maryland, College Park, MD (2005)

[18] Davuluri C. Role-Based Access Control in Collaborative Research Environments. Thesis. Dept. of Computer Science and Engineering, The Ohio State University. (2010)