Collaborative Management of Web Ontology Data with Flexible Access Control

Jie Lu[a], Chao Wang[a], Guangquan Zhang[a], Jun Ma[a]

[a]*Faculty of Engineering and Information Technology*
University of Technology, Sydney (UTS)
PO Box 123, Broadway, NSW 2007, Australia

Abstract

The creation and management of ontology data on web sites (e.g. instance data that is used to annotate web pages) is important technical support for the growth of the semantic web. This study identifies some key issues for web ontology data management and describes an ontology data management system, called *robinet*, to perform the management. This paper presents the structure of the system and introduces a Web ontology data management model that enables a flexible access control mechanism. This model adds rules into the *robinet* system to utilise the semantics of ontology for controlling the access to ontology data. The implementation of the rule-based access control mechanism and related testing are also discussed.

Key words: Collaborative management, Web ontology data, Access control

1. Introduction

Ontologies have been recognized as an essential layer of the emerging semantic web [5, 20]. As a result, the abundance of ontology related information is critical to the maturity of the semantic web. Through the semantic web search engine Swoogle [11], we have found that there are plenty of ontology schemas (which refer to classes, properties and their relations, such as "TBox" in description logics [4]) available over the web. By contrast, ontology data or instance data (which refer to instances of classes or individuals such as those contained in "ABox") do not seem to be very abundant. One of the barriers for the problem is that it is not easy for ordinary users to create ontology data due to the complexity of ontology languages compared to the creation of normal web pages. Several tools have been developed to assist the generation of ontologies with ease (e.g., Protégé [17], OntoEdit [37], SWOOP [26]), but most are not web-based tools and are not often used by ontology experts due to their complexity. In addition, while ontology schemas, which usually reflect the domain knowledge, are relatively stable once developed, the ontology data is often prone

Email addresses: jielu@it.uts.edu.au (Jie Lu), cwang@it.uts.edu.au (Chao Wang), zhangg@it.uts.edu.au (Guangquan Zhang), junm@it.uts.edu.au (Jun Ma)

to changes in a dynamic environment. Therefore, it may not be convenient to use offline tools to maintain the ontology data.

Recently, wikis[1] have been studied as an alternative way to create and maintain ontology data on the web. The basic idea of most wiki systems is to let web users create and maintain web contents in a collaborative way. Semantic wiki systems additionally introduce semantic web standards (RDF/RDFS [29, 6], OWL [30]) and techniques to ordinary wiki systems, trying to ensure the collaborated contents incorporate more semantics for computers to understand and process. However, when collaboration is introduced into the management of web ontology data, some particular issues need to be tackled properly, such as the reliability of the data and the security of the management systems. Adequate access control mechanisms are often needed to handle these issues.

This paper discusses the collaborative management of web ontology data with a focus on the enforcement of a flexible access control mechanism. It first analyzes the functions that are desirable for a management system. Accordingly, some key issues such as usability, data reliability and data quality are identified. A web ontology data management system framework (on which the *robinet* system is based) is described to accommodate these desirable functions. This paper also proposes an ontology data management model that underlies the system framework and discusses the implementation of this model. In particular, the flexible access control mechanism that utilizes both the rules and embedded semantics of ontology data is presented. Discussions on the comparisons of this design paradigm and other paradigms for web content management are also given.

The rest of the paper is organized as follows. Section 2 reviews related work in the fields of semantic wikis, data-intensive applications, and access control over the web. Section 3 analyzes the desirable functions for managing ontology data and presents the framework of the proposed web ontology data management system. Section 4 introduces a web ontology data management model that enables the flexible access control mechanism. Section 5 presents the implementation details of the model including some key operational functions and the access control mechanism. Section 6 discusses some main advantages and limitations of the proposed web ontology data management system. Section 7 concludes the paper and proposes future work.

2. Related Work

Our research is related to the topics of wiki systems, data-intensive web applications and access control over the web. This section first reviews the general ideas of wiki systems and the development of typical semantic wiki systems. Related work in data-intensive web applications and access control over the web is also discussed.

[1] http://en.wikipedia.org/wiki/Wiki

2.1. Semantic Wikis

Most wiki systems are designed to let web users create and maintain web contents in a collaborative way. Most existing systems provide convenient functions so that an ordinary user with little knowledge of web page design and creation can contribute contents to the web. Wikipedia[2], the most well-known online collaborative encyclopedia on the web, is a successful application of wiki systems. However, most contents maintained in current wiki systems are semi-structured web pages only for web users to search and read. The facts and knowledge expressed in those web pages lack proper syntax and semantics for computers to process.

In order to deal with this issue, semantic wiki systems have emerged to introduce semantic web standards (e.g., RDF/RDFS , OWL) and techniques to the common wiki systems, and enable the collaborated contents to include more semantics for computers to understand and process.

Platypus Wiki [38] is an extended wiki system that allows users to input RDF and OWL statements in addition to plain text. How the system helps users to create such semantic statements has not been discussed well in [38]. Rhizome [35] is an application stack that lets developers build Web applications which use familiar technology such as XML. It can be used to develop a wiki-like application that lets the user create arbitrary RDF resources and the application logic for presenting and interacting with them. Several custom text formats (e.g., ZML, RxML) are used in the system to write semantic content. The use of those custom text formats may help experienced users to generate semantic contents effectively and allow systems to process them efficiently, but it can also raise barriers for ordinary users.

Powl [1] is a web based platform that provides a collaborative environment for semantic web development. OntoWiki [2], which is built upon Powl, employs web 2.0 techniques to bring more features in authoring, editing, and searching semantic contents. Unlike Powl and OntoWiki, which concentrate more on ontology editing, Semantic Wikipedia [41], as an extension to the popular MediaWiki software, brings semantics to texts and links in its wiki system. These systems usually have more sophisticated user interfaces.

There exist other semantic wiki systems (e.g., WikSAR [3], COW [13], WikiFactory [22]) which provide more features such as integration with desktops [3], typical query mechanisms [13], domain-oriented design [22], and so on. The investigation shows that most wiki systems do not have a sophisticated mechanism to control the operations on the contents. The lack of this mechanism may reflect a particular feature of wikis, that is, an open access for a community. However, while the use of wikis has been widespread from loosely organized web communities to well organized companies, it is desirable that a flexible access control mechanism be an option for controlling the operations of wiki contents in certain situations.

[2]http://www.wikipedia.org

2.2. Data-intensive Web Applications

The research in data-intensive web development is motivated by the need to make a large volume of data available on web sites for people to access. Various tools and approaches have been proposed in this research area [14].

Araneus [31], as a web-based management system, is able to manage both structured and semi-structured data in a database style. It supports three main classes of applications (or operations) in the spirit of databases, which are "queries", "views", and "updates". Restrictions on these operations are not yet well discussed. Strudel [12] is a system aimed at separating the management of a web site's data, specification of its contents and structure, and presentation of its pages. It provides a declarative query language and a template language to achieve this goal. Autoweb [15] is another system that relies on a conceptual model and a tool environment to support web site development. All these systems employ database and/or XML techniques to perform conceptual modeling and use particular languages for specification and transformation. Mostly, they follow a centralized approach to generate, control and maintain the data in web applications. As a result, the generated web sites to most users are read-only online data repositories that allow limited interaction.

OntoWebber [24] tries to build a data-intensive web site not only by explicit modeling of different aspects of the web site, but also by using ontologies as the foundation for web site design. The management of data in the system (e.g., site maintenance modeling) still adopts a centralized mode of operation. An administrator is responsible for almost all the content maintenance tasks, and other users are just content consumers of the web site.

Recently, the need for interaction between data-intensive web applications and their users has been addressed in [23], which proposes an "operation model" in addition to modeling contents, navigation and presentation. XML techniques are used to help generate an operation layer in the application from the operation model. However, the paper does not provide more information on how such operations are restricted in the system to prevent unwanted operations to the contents from unauthorized users.

2.3. Access Control and Security over the Web

Since this study tackles privacy and security issues in managing web ontology data, it is related to the topic of access control.

One of the most well-known approaches to access control is the use of a family of role-based access control (RBAC) models [34], in which different permissions are associated with roles and users are assigned to appropriate roles. Due to its simplicity and flexibility, this approach has been used in modern operating systems and database management systems [34]. Since roles are normally predefined with different permissions, permissions are fixed once a user is assigned a given role. It is then difficult to adjust the user's permissions unless the user is assigned to a different role, or the role is modified directly. Therefore, RBAC models may be used well in situations where the roles have already been clearly defined and are stable for a period of time. In addition, they do not have any inherent support for fine grained authorization [33].

Along with RBAC models, other access control approaches have been studied in the database research community [8]. One approach is to use views as the basis for access control. In particular, authorization-transparent access control models using views with validity inference have been proposed [33], which allow fine grained access control levels other than the level of tables and columns. The use of an access matrix model [19] is another approach to access control in databases. Since these approaches are studied mainly in databases, they normally require that controlled data is structured into tables with certain normalization.

Policy-based approaches are another family of methods increasingly used for automated system management [39]. Several policy languages and frameworks, such as Ponder [10] , Rei [25], KAoS [40], and extended XACML [9], have been proposed, and some of them have employed semantic web technologies (i.e., ontology, description logics (DLs), and rules) to model, represent and inference on policies.

There are some systems that have their own policy/rule languages for access control and privacy issues. For example, the Semantic e-Wallet project [16] uses rules represented in RDF-S/OWL and translates them into a format that can be processed by JESS, a Java-based rule engine. One of its objectives is to allow users to control accessibility of their contextual information to selected users.

In summary, RBAC models are simple to implement, but may not be applicable for dynamic situations that require fine grained access controls. These approaches to access control in databases may be efficient in dealing with a large volume of data, but they usually require structured data. It is possible to either use a general existing policy language and its framework or to create a new special mechanism to handle the issue of access control for a web ontology data management system. We prefer to create a special mechanism because using a general existing policy language/framework often requires additional modeling work that bridges the domain of the system and the used policy language/framework. This may make the policies less intuitive and hard to understand. Therefore, further optimization of them may not be possible. Compared to role-based models, our mechanism is more adapted to dynamic situations. In addition, it allows the achievement of fine grained access control in an incremental way without the requirement of database-like structured data.

3. Design of the Web ontology Data Management System

3.1. Desirable Functions for Managing Web Ontology Data

This study identifies four sets of desirable functions that could be required for managing web ontology data and provides some hints towards the design and development of more extensible semantic wiki systems.

Authentication and access control (**A**). This set of functions deals with the identification of the users and the control of their operations on the web ontology data. Existing semantic wiki systems seem to lack a sophisticated mechanism to implement this type of function. The lack of this mechanism affects the ability to trust the contents from people, which hinders their use in some serious situations. Two interesting examples are:

(1) an IT company wants to have the information of its employees, divisions, projects, documents and so on managed collaboratively through a web based platform but hopes to maintain a certain level of security and privacy;

(2) a department at a university requires that its web site can be updated by its staff, professors and students so that the web site reflects the latest information about professors (contacts, publications, courses to be offered, etc), students (contacts, courses enrolled in) and other resources.

Browsing, creating and editing (**B**). This set of functions involves the provision of facilities and interfaces for browsing the ontology schema and data, creating and updating the data as required. These functions are actually the most fundamental functions for any type of semantic wikis. Since the formats of ontology data (e.g., OWL) are not intuitive to ordinary users, it is desirable to design these functions with more intuitive and interactive features.

Search and Query (**S**). Obviously, these functions are designed to retrieve created ontology data from web sites in response to the information need of the users. Generally, the search function may refer to some form of query upon free/unstructured texts such as the search provided by web search engines. In contrast, the query function is usually associated with the structured data (i.e., SQL for the relational databases). Since web ontology data is a type of semi-structured data, it is desirable to have both search and query functions in semantic wikis.

Data quality control (**D**). Generally, data quality control can be achieved in non-technical ways (e.g., training the authors to publish high quality data, or establishing peer review mechanisms to improve data quality). In addition to these non-technical ways, some technical functions can also help ensure that the overall web ontology data contributed by users is of accepted quality. For example, a common issue of data quality is that the duplicated data may be generated in a web environment. A proper use of deduplication techniques can help to deal with this data quality issue.

3.2. The Robinet System Framework

This sub-section overviews a framework based on which the prototype system "*robinet*" is developed. It shows the general structure of the framework, the components (and their relationships) that reflect the desirable functions previously discussed. The *robinet* system is implemented in Java.

Figure 1 illustrates the general structure of the framework and the interaction between the framework and its managed web ontology data and users. At its lowest layer, the data layer, access rules and indexed data are stored in addition to the managed ontology data. Above the data layer is the API layer, which is made up of low level packages and the APIs they offer. Based on these APIs, a supporting layer provides customized tools which are used to build various operations, data models and functions for presentation.

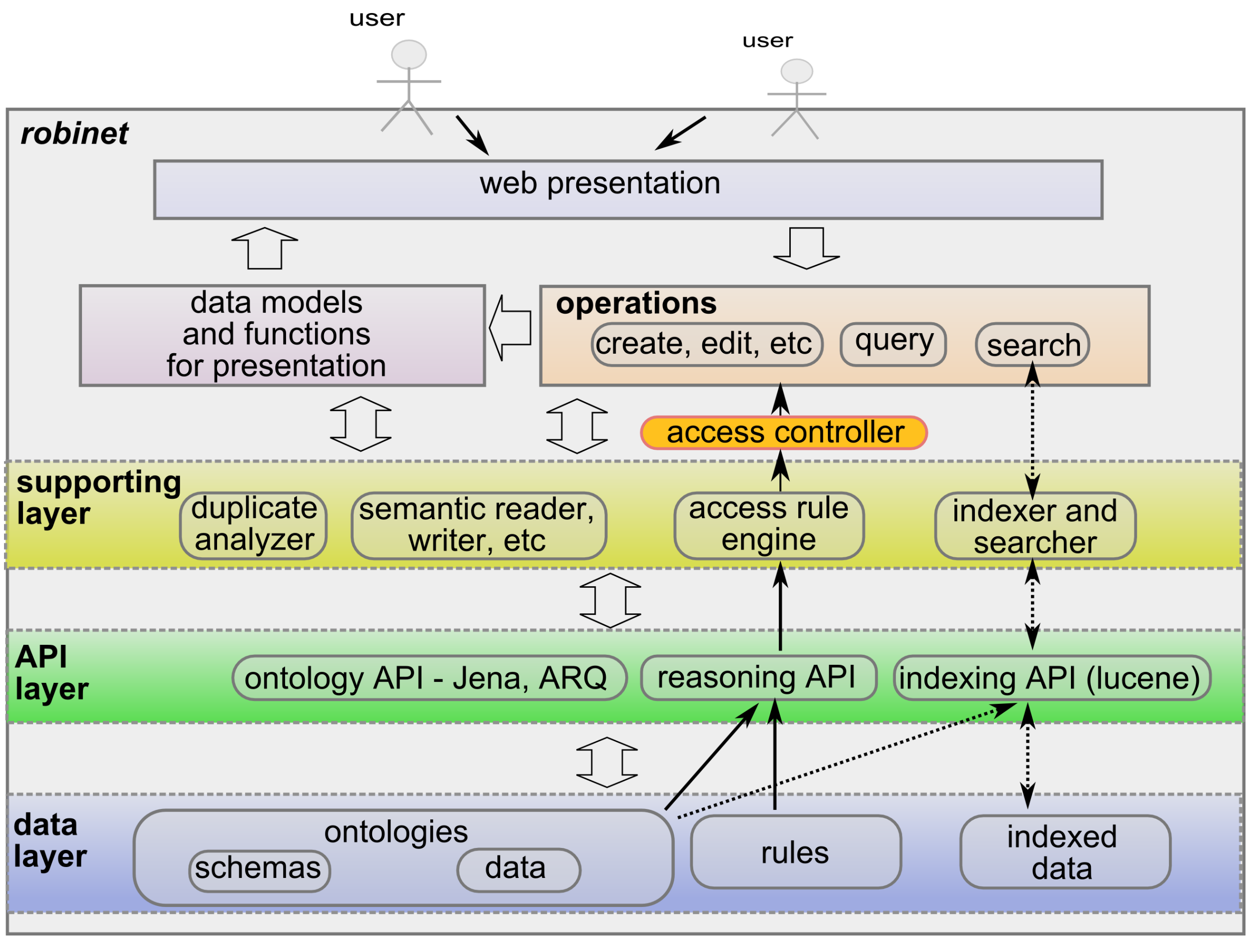


Figure 1: A framework for the prototype system *robinet*

The four sets of desirable functions (A, B, S, D) mentioned in Section 3.1 are reflected in the illustrated framework. Basic and common functions such as browsing, creating and editing are implemented through certain operations. These operations are restricted by an "access controller", which relies on an "access rule engine" to make decisions to accept/reject operations, avoiding malicious editing or sensitive data leaking in certain situations. The "indexer and searcher" and their related components are designed for the implementation of the search functions on ontology data. The "duplicate analyzer" is responsible for the detection of duplicated ontology data, thus helping improve the data quality. This paper mainly discusses in detail the first two functions in the framework, which are common operations (browsing, creating and editing) and flexible access control. Discussion on the other functions can be found in [44]. In addition, the methods that can be used to implement the "duplicate analyzer" have been proposed in [42][43].

Besides these functions, the following main principles are also considered in developing the *robinet* system.

Modularity. This has become a common practice in software development. Guided by this principle, we try to divide the system into several loosely coupled modules and group them into layers, as revealed by

Figure 1. This principle allows us to modify or update a module with little interference to other modules.

Friendly URLs. Unlike some web sites that use very long and strange URLs for their web contents, wiki systems often use URLs that are both user and search engine friendly. This feature has become a trend nowadays. For the semantic web, this is not just a fashion, as a friendly URL is often a permanent URL that is used to identify resources on the semantic web. Therefore, the *robinet* system uses friendly URLs, particularly for concepts and instances. Section 5.1 will illustrate how we follow this principle.

Ease of use. The *robinet* system aims to provide enough facilities to make ordinary users feel comfortable when they use it.

Before describing the detailed implementation of the functions and the demonstration of the principles, we first present the underlying model for web ontology data management.

4. Web Ontology Data Management Model

We provide a web ontology data management model in order to impose flexible access control of the data. At a certain level of abstraction, a web ontology data management system can be modeled as $\Lambda(C, O, P, R)$ where C is the set of data instances of ontology concepts (classes), O the set of operations on semantic contents, P the set of different types of users who participate in the system by contributing and sharing contents through the set of operations O, and R the set of rules that control the operations in O performed by participants P on the semantic contents C. Formally, the set of R is a set of mappings from $P \times C$ to O.

4.1. Semantic Contents

Semantic contents are composed of data instances of ontology concepts (classes). Before semantic contents are created and managed, we need to have one or more ontology schemas containing proper concepts and relations for a given domain. Therefore, a proper ontology schema is an important prerequisite. A well modeled ontology schema casts rich semantics to the ontology data, which helps users to understand the ontology data well. Ontology schema modeling can be conducted through ontology engineering [36]. There exist several supporting tools (e.g., Protégé [17]) and approaches (such as ontology learning [28]) for this task. This study assumes that proper ontology schemas are available for annotating semantic contents.

To assist further discussion in the following sections, we give an example of some fragments of ontology schema reflecting the domain of company information management as shown in Figure 2 (referred to as $S_{company}$ in subsequent sections). They are defined with the notations from Description Logics. Given this example, an instance of the concept `Developer` can be denoted as `Developer(john)`, where `john` identifies the instance. All these instances then form the semantic contents for the given domain or the specific application area.

4.2. Participants

Since participants represent the set of different types of users of the system, they can be modeled into ontology schemas in terms of types and relations. In particular, for some domains, they form an essential part that is closely related to other aspects of the domains. For example, in the ontology schema $S_{company}$ that describes the structures of the company and its resources, participants are mostly employees, whose types and relations are already modeled into $S_{company}$.

When participants are modeled in ontology schemas (whether in domain ontologies or in separate ontologies) and these ontology schemas form the basic schemas for this system, the instances of different types of participants actually become the contents that are manageable through the system. For example, while John is a participant with the role of `Developer`, the instance `john` that represents John becomes semantic contents `editor` under management.

$$
\begin{aligned}
\texttt{Company} &\sqsubseteq \texttt{Organization} \\
\texttt{Division} &\sqsubseteq \exists\texttt{partOf.Organization} \\
\texttt{Group} &\sqsubseteq \exists\texttt{partOf.Division} \sqcap \exists\texttt{member.Person} \\
\texttt{Employee} &\sqsubseteq \texttt{Person} \sqcap \exists\texttt{workFor.Company} \\
\texttt{Manager} &\sqsubseteq \texttt{Employee} \\
\texttt{ProjectManager} &\sqsubseteq \texttt{Manager} \sqcap \exists\texttt{manage.Project} \\
\texttt{FinancialStaff} &\sqsubseteq \texttt{Employee} \\
\texttt{Developer} &\sqsubseteq \texttt{Employee} \sqcap \exists\texttt{memberOf.Group} \\
\texttt{Report} &\sqsubseteq \texttt{Document} \\
\texttt{ProjectDocument} &\sqsubseteq \texttt{Document} \sqcap \exists\texttt{belongTo.Project} \\
\texttt{ProjectReport} &\sqsubseteq \texttt{Report}\sqcap\exists\texttt{belongTo.Project} \\
\texttt{Specification} &\sqsubseteq \texttt{ProjectDocument}
\end{aligned}
$$

Figure 2: Fragments of an ontology schema reflecting the domain of company information management

4.3. Operations

Operations form the interaction between the semantic contents and the participants. There are four basic operations, “create”, “edit”, “delete”, and “view”, in web ontology instances. For instance, the “view” operation can be regarded as an operation that retrieves an instance and presents it to users in a specificway. For different applications or domains, concrete meanings will be associated with them. In addition, there can be more specific operations with typical semantics. For example, in a blog application, we may have a “comment” operation on the articles published by bloggers. In the domain of company information management, we may have a “change salary” operation on the instance of employees. Such operations can be viewed as special types of the four basic operations. In a complex system where many different types of specific operations are developed, it is desirable to use an “operation ontology” to depict them.

4.4. Rules

Rules are used to control the operations of participants on the semantic contents. As mentioned before, a rule defines a mapping from $P \times C$ to O. Informally, rules can be described using intuitive "if-then" statements. For example, a simple rule may be informally described as "if a participant is with the type of `Administrator`, then he/she can perform any operation on any instance", while a more complex rule may look like "if a participant is of type of `Developer`, then he/she can view and edit specifications that belong to the project he/she works on". Unlike operations and participants, rules are not modeled into ontology schemas in our method. This is partly because ontology languages, which are based on description logics, have several limitations [32]. Adding a rule layer on top of ontology is a common practice and efforts have been made to define rule languages for the semantic web such as SWRL [21]. By using rules, it is possible to use logic programming (LP) combined with description logics [18] to perform reasoning. This study uses Prolog to implement the rule inference, which is discussed in detail later.

The following are some examples of rules to control operations in regard to $S_{company}$.[3]

$$\texttt{Employee}(?x) \Rightarrow \texttt{edit}(?x, ?x) \tag{1}$$

$$\texttt{Developer}(?x) \wedge (\exists y)\texttt{workOn}(?x, y) \Rightarrow \texttt{create}(?x, \texttt{Specification}) \tag{2}$$

$$\begin{aligned}&\texttt{Developer}(?x) \wedge \texttt{Specification}(?y) \wedge \texttt{belongTo}(?y, ?z) \wedge \texttt{workOn}(?x, ?z)\\ &\qquad \Rightarrow \texttt{edit}(?x, ?y)\end{aligned} \tag{3}$$

These rules are expressed in an intuitive way for demonstration. For example, Rule (1) states that any employee can edit contents of his/her instance. Rule (3) states that a developer can edit a specification if he/she works on the project to which the specification belongs. During implementation, these rules are recoded using Prolog clauses, which will be discussed later.

There are two types of rules: permission and rejection (similar to the policy constructs "rights" and "prohibitions" in some policy languages [10, 25]). For example, Rule (4) is a rejection rule that does not allow the creation of any instances of `Employee` by developers.

$$\texttt{Developer}(?x) \Rightarrow \neg\texttt{create}(?x, \texttt{Employee}) \tag{4}$$

We are able to define very flexible restrictions on operations thanks to the reasoning capabilities of ontologies together with rules. For example, given the ontology schema $S_{company}$, we can obtain Rule (5) indicating that a developer is an employee. Then we know that a developer can also edit his/her own instance using Rule (1).

$$\texttt{Developer}(?x) \Rightarrow \texttt{Employee}(?x) \tag{5}$$

[3] `edit`$(?x, ?x)$ may look somewhat confusing. We explain that the first $?x$ indicates the employee described by the instance while the second $?x$ represents the contents of the instance.

In summary, rules provide a flexible way to impose access control in the model by using the rich semantics provided by the underlying ontology. However, when more rules are added, the relationships between these rules may become complicated. Rule conflict issues may arise. They will be addressed in the implementation of the rule engine, presented later in the paper.

5. Model Implementation

This section discusses the implementation of the model in terms of some desirable key operations and the access control mechanism.

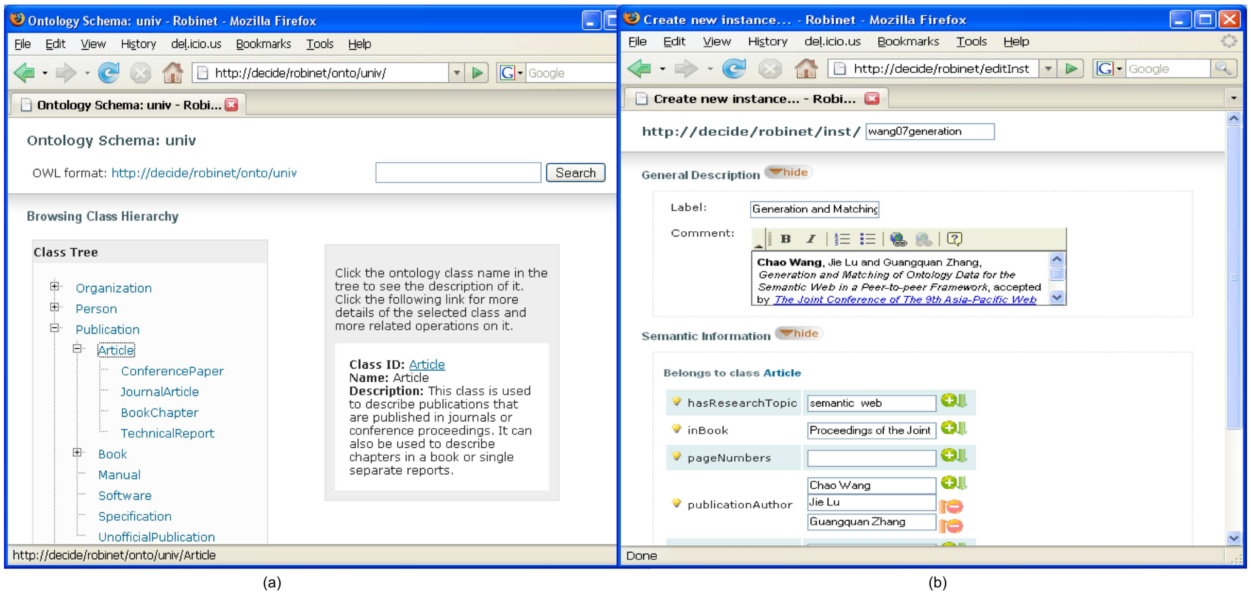
5.1. Key Operations

5.1.1. Ontology Schema Browsing

Browsing ontology schemas helps ordinary users become familiar with the domain that the ontology schemas are intended to describe. It is clear that a user without an adequate understanding of the domain ontology schema may not be able to proceed and create correct instances for the domain. Some ontology specific editing tools (e.g., Protégé [17]) provide graphical user interfaces that allow users to view the domain ontology schemas intuitively. But this requires users to install additional software. In addition, since such editing tools are mostly designed for domain experts or certain professionals to develop ontologies in a desktop environment, it may not be as easy to use as web browsers which are commonly used by ordinary users nowadays. Therefore, it is desirable to have a web-based browsing mechanism so that users can use any type of browser to learn the domain ontology schemas before they are ready to contribute information.

Initially, the *robinet* system requires a setup process such as loading the ontology schemas for browsing. Once the initial setup is done, users can view the ontology schemas using common browsers. Figure 3 (a) shows the web interface which renders the classes (concepts) in an ontology schema about the academic domain in a tree structure according to their 'is-a' relationship. Users can get familiar with these classes and their relations by exploring this tree. In addition, if they are interested in a particular class, they can click it in the tree to see detailed comments (if supplied by the schema makers) on this class in the right-hand side panel. If further information is needed about this class, users can follow the link in the right-hand side panel to view such information.

In the *robinet* system, the URLs used to identify their corresponding resources (classes or properties) are quite friendly to both end users and other semantic web applications which may run on other computers in the network. For example, while the URLs `http://decide/robinet/onto/univ/` and `http://decide/robinet/onto/univ/Article` lead to human-readable web pages about the ontology schema "`univ`" and the its class "`Article`" respectively, the URLs `http://decide/robinet/onto/univ` and `http://decide/robinet/`
identify the original OWL formatted versions which are more machine-understandable. Therefore, other semantic web applications can use such URLs to remotely import the interested ontology schemas into their


Ontology Schema: univ - Robinet - Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help
http://decide/robinet/onto/univ/
Ontology Schema: univ - Robi...
Search
Class Tree
Organization
Person
Publication
Article
ConferencePaper
JournalArticle
BookChapter
TechnicalReport
Book
Manual
Software
Specification
UnofficialPublication
(a)
Create new instance... - Robinet - Mozilla Firefox
http://decide/robinet/editInst
Create new instance... - Robi...
hide
B I
Done
(b)

Ontology Schema: univ

OWL format: http://decide/robinet/onto/univ

Browsing Class Hierarchy

Click the ontology class name in the tree to see the description of it. Click the following link for more details of the selected class and more related operations on it.

Class ID: Article
Name: Article
Description: This class is used to describe publications that are published in journals or conference proceedings. It can also be used to describe chapters in a book or single separate reports.

http://decide/robinet/onto/univ/Article

http://decide/robinet/inst/ wang07generation

General Description

Label: Generation and Matching

Comment: **Chao Wang**, Jie Lu and Guangquan Zhang, *Generation and Matching of Ontology Data for the Semantic Web in a Peer-to-peer Framework*, accepted by *The Joint Conference of The 9th Asia-Pacific Web*

Semantic Information

Belongs to class **Article**

hasResearchTopic	semantic web
inBook	Proceedings of the Joint
pageNumbers	
publicationAuthor	Chao Wang
	Jie Lu
	Guangquan Zhang

Figure 3: The interfaces for browsing domain ontology schema and editing ontology data

application space. This mechanism could help the development of semantic web applications and intelligent software agents over the web.

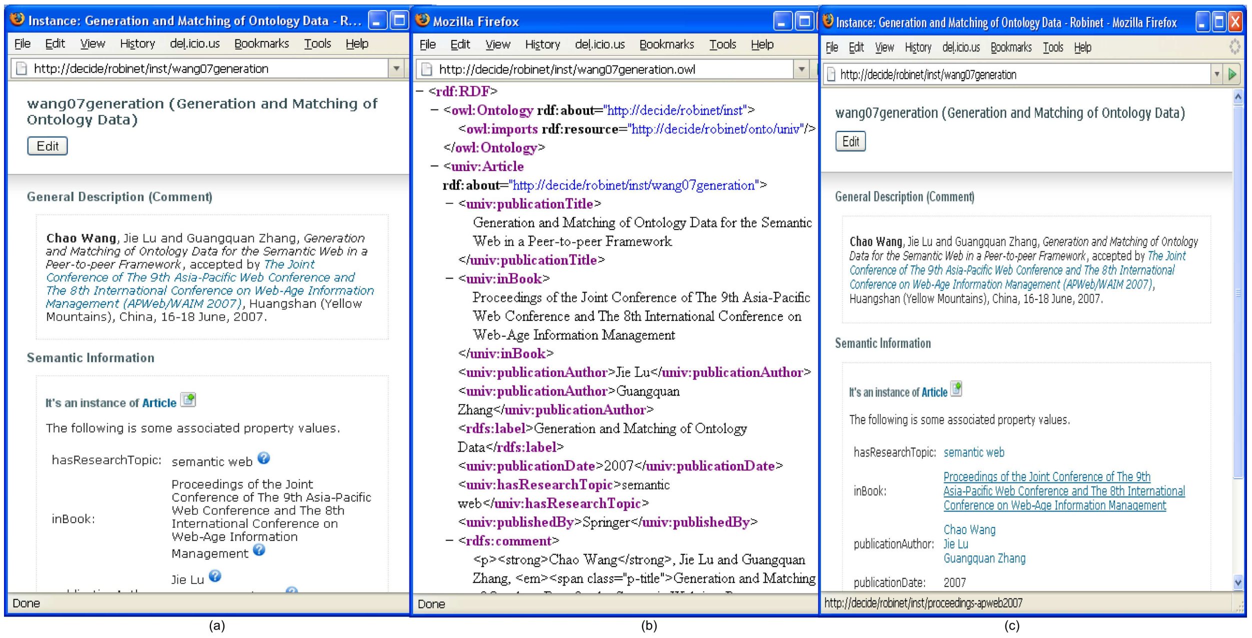
5.1.2. *Ontology Data Creation and Editing*

Once users have become familiar with the ontology schemas by browsing them, they can create new ontology data through the functions of the system. Currently, two main types of functions (*new data creation* and *related data creation*) have been implemented.

New data creation enables users to create completely new ontology data. For example, a staff member who wants to publish some data about their recent publications, can choose the class that is appropriate for the data. They may choose the class "`Article`" in the ontology schema "`univ`" and use it as the type for the new instance created to annotate a paper. Figure 3 (b) demonstrates the interface used for this operation. A unique ID is required from the user to form the URL used to identify this publication.

Figure 4 (a) shows the created instance in a human-readable view, which can be accessed through the URL generated, based on the user-supplied ID. Actually, this view is generated by the system from the OWL formatted ontology data which can also be accessed through a browser, as shown in Figure 4 (b). Obviously, the actual ontology data is not easy for ordinary users to read. Computers, on the other hand, are good at processing such data. Since they are aligned with the given ontology schema, well designed semantic web applications that are aware of this domain ontology can use them to perform further complicated tasks such as classification and reasoning.

After instances are created, they can be further edited by clicking the "Edit" button, as shown in Figure 4 (a).



Instance: Generation and Matching of Ontology Data - R...

http://decide/robinet/inst/wang07generation

wang07generation (Generation and Matching of Ontology Data)

General Description (Comment)

Chao Wang, Jie Lu and Guangquan Zhang, *Generation and Matching of Ontology Data for the Semantic Web in a Peer-to-peer Framework*, accepted by *The Joint Conference of The 9th Asia-Pacific Web Conference and The 8th International Conference on Web-Age Information Management (APWeb/WAIM 2007)*, Huangshan (Yellow Mountains), China, 16-18 June, 2007.

Semantic Information

It's an instance of **Article**

The following is some associated property values.

hasResearchTopic:	semantic web
inBook:	Proceedings of the Joint Conference of The 9th Asia-Pacific Web Conference and The 8th International Conference on Web-Age Information Management
	Jie Lu

(a)

Mozilla Firefox

http://decide/robinet/inst/wang07generation.owl

```
- <rdf:RDF>
  - <owl:Ontology rdf:about="http://decide/robinet/inst">
      <owl:imports rdf:resource="http://decide/robinet/onto/univ"/>
    </owl:Ontology>
  - <univ:Article
    rdf:about="http://decide/robinet/inst/wang07generation">
    - <univ:publicationTitle>
        Generation and Matching of Ontology Data for the Semantic
        Web in a Peer-to-peer Framework
      </univ:publicationTitle>
    - <univ:inBook>
        Proceedings of the Joint Conference of The 9th Asia-Pacific
        Web Conference and The 8th International Conference on
        Web-Age Information Management
      </univ:inBook>
      <univ:publicationAuthor>Jie Lu</univ:publicationAuthor>
      <univ:publicationAuthor>Guangquan
      Zhang</univ:publicationAuthor>
      <rdfs:label>Generation and Matching of Ontology
      Data</rdfs:label>
      <univ:publicationDate>2007</univ:publicationDate>
      <univ:hasResearchTopic>semantic
      web</univ:hasResearchTopic>
      <univ:publishedBy>Springer</univ:publishedBy>
    - <rdfs:comment>
        <p><strong>Chao Wang</strong>, Jie Lu and Guangquan
        Zhang, <em><span class="p-title">Generation and Matching
```

(b)

Instance: Generation and Matching of Ontology Data - Robinet - Mozilla Firefox

http://decide/robinet/inst/wang07generation

wang07generation (Generation and Matching of Ontology Data)

General Description (Comment)

Chao Wang, Jie Lu and Guangquan Zhang, *Generation and Matching of Ontology Data for the Semantic Web in a Peer-to-peer Framework*, accepted by *The Joint Conference of The 9th Asia-Pacific Web Conference and The 8th International Conference on Web-Age Information Management (APWeb/WAIM 2007)*, Huangshan (Yellow Mountains), China, 16-18 June, 2007.

Semantic Information

It's an instance of **Article**

The following is some associated property values.

hasResearchTopic:	semantic web
inBook:	Proceedings of the Joint Conference of The 9th Asia-Pacific Web Conference and The 8th International Conference on Web-Age Information Management
publicationAuthor:	Chao Wang Jie Lu Guangquan Zhang
publicationDate:	2007

http://decide/robinet/inst/proceedings-apweb2007

(c)

Figure 4: The created instance of the class `Article`. (a) The instance shown in a human readable view; (b) The actual ontology data created by the user. (c) The instance after related instances are created and automatically linked with it.

Related data creation allows users to create new ontology data which is related to the existing data.

First, we discuss why we need this type of function. The function relies on the existence of a type of properties, called "object properties", which link one instance to another with a particular semantic relation [30]. It offers the functionality similar to hyperlinks which are used over the current web to link different web pages. The importance of hyperlinks is obvious. Without them, the web could not form and grow into the huge connected information repository, nor could modern search engines build a large information base and rank the related information with certain measurements such as authority or popularity [27][7]. Similarly, it is also very important to use object properties to link instances. The resulting relational feature in ontology data could be even more essential than using normal hyperlinks in web pages, as it delivers more specific semantic meanings than a normal hyperlink mechanism. Therefore, it is highly desirable to enable users to create ontology data which is linked to other ontology data via certain object properties.

This system then provides an easy way to create related new ontology data based on existing ontology data. As already explained, when a completely new instance is created, all property values are initially supplied as strings by users, regardless of the types of properties. However, after the instance is created, the system will detect whether a property value should be a certain data type or an instance according to the ontology schema. For those that should be instances via given object properties, it provides the function to create them. As shown in Figure 4 (a), those property values that should be instances have associated with question marks. By clicking a question mark, users will be presented with an instance creation interface which is similar to the one shown in Figure 3 (b). After creating these new instances, they are related to the existing instance with the specified object properties. Figure 4 (c) shows the web pages of the existing instance after its related instances are created. The question marks have disappeared. Instead, those object property values are linked to the web pages of their related instances.

5.2. Implementation of Access Control

The functionality of access control in the system is mainly controlled by the rule engine, which has been implemented with Prolog. The knowledge reflected by the domain ontology schema is input into a Prolog database for reasoning. For example, the "*is-a*" relation between the concepts `Developer` and `Employee` is asserted as a clause:

$$\texttt{c\_Developer}(X) :- \texttt{c\_Employee}(X). \tag{6}$$

If a new instance of `Developer` "John" is created, it will be asserted as a fact:

$$\texttt{c\_Developer}(\texttt{i\_John}). \tag{7}$$

As mentioned before, rules are re-coded into Prolog clauses and their priorities are also considered. The head of the rule clause is a 5-arity functor "accept" or "reject", representing the two types of rules. The five arities are (*P, O, C, Pr, PropSet*), where *P, O, C* represent participants, operations, semantic contents (ontology data) respectively, *Pr* represents the priority of the rule, and *PropSet* represents the affected property set associated with the targeted ontology data. For example, the rule clause (8) means that a developer can edit all the properties (indicated as "all" in the clause) of a specification when he/she works on the project to which the specification belongs. This actually implements Rule (3). This rule has a priority of 2 and can be overridden by rules with priorities higher than 2.

$$\begin{aligned} &\texttt{accept}(P, \texttt{edit}, C, 2, \texttt{all}) :- \\ &\quad \texttt{c\_Developer}(P), \texttt{c\_Specification}(C), \texttt{p\_belongTo}(C, T1), \texttt{p\_workOn}(P, T1). \end{aligned} \tag{8}$$

5.2.1. Rule Priority

If a large number of rules is defined to control operations, then conflicts between rules can hardly be avoided. Rule priority is used to partly resolve rule conflicting. As shown in the "accept/reject" functor, we assign a number between 0 and 10 to *Pr* as the priority to each rule clause. When two rules are found to conflict with each other, the rule with higher priority will override the other. For example, Rule (9) states that any operations on any contents are rejected with the priority of 0, the lowest priority, while Rule (10) accepts the "view" operation on any contents with the priority of 1. Therefore, the "view" operation will be always acceptable, despite of the presence of Rule (9).

$$\texttt{reject}(P, O, C, 0, \texttt{all}) :- \texttt{operation}(O), \texttt{content}(C). \tag{9}$$

$$\texttt{accept}(P, \texttt{view}, C, 1, \texttt{all}) :- \texttt{content}(C). \tag{10}$$

Sometimes, conflicts may still exist when two conflicted rules are given the same priority. When such conflicts are detected, the system will report the issue and discontinue the operations. In this case, human intervention is required to resolve conflicting rules before any permissions may be granted to the requested operations.

5.2.2. Property Level Access Control

Property level access control allows access control to be imposed at the property level, which is more fine-granular and flexible. The requirement of this level of access control is common in practice. For example, it is desirable that the salary (or other sensitive information) of an employee is only viewable by certain participants including the employee and financial staff, while the basic information of the employee can be viewable by all participants. In the *robinet* system, this can be implemented using Rules (11) and (12). Both of the rules have the same priority level, therefore, they cannot override each other. This could lead to a conflict if property level access control is not considered. However, when property level access is considered, the set of accepted properties associated with the instance will be computed through the values of *PropSet* from the effective rules. So in this example, the value of property `hasSalary` associated with the `Employee` instance is only viewable by the employee himself and the financial staff while other property values can be viewed publicly, as shown in Figure 5.

$$\begin{aligned} &\texttt{accept}(P, \texttt{view}, C, 1, \texttt{all}) :- \\ &\quad \texttt{c\_Employee}(C). \end{aligned} \tag{11}$$

$$\begin{aligned} &\texttt{reject}(P, \texttt{view}, C, 1, [\texttt{p\_hasSalary}]) :- \\ &\quad \texttt{c\_Employee}(C), P \backslash = C, \texttt{not}(\texttt{c\_FinancialStaff}(P)). \end{aligned} \tag{12}$$

5.2.3. Rule Coverage and Engineering

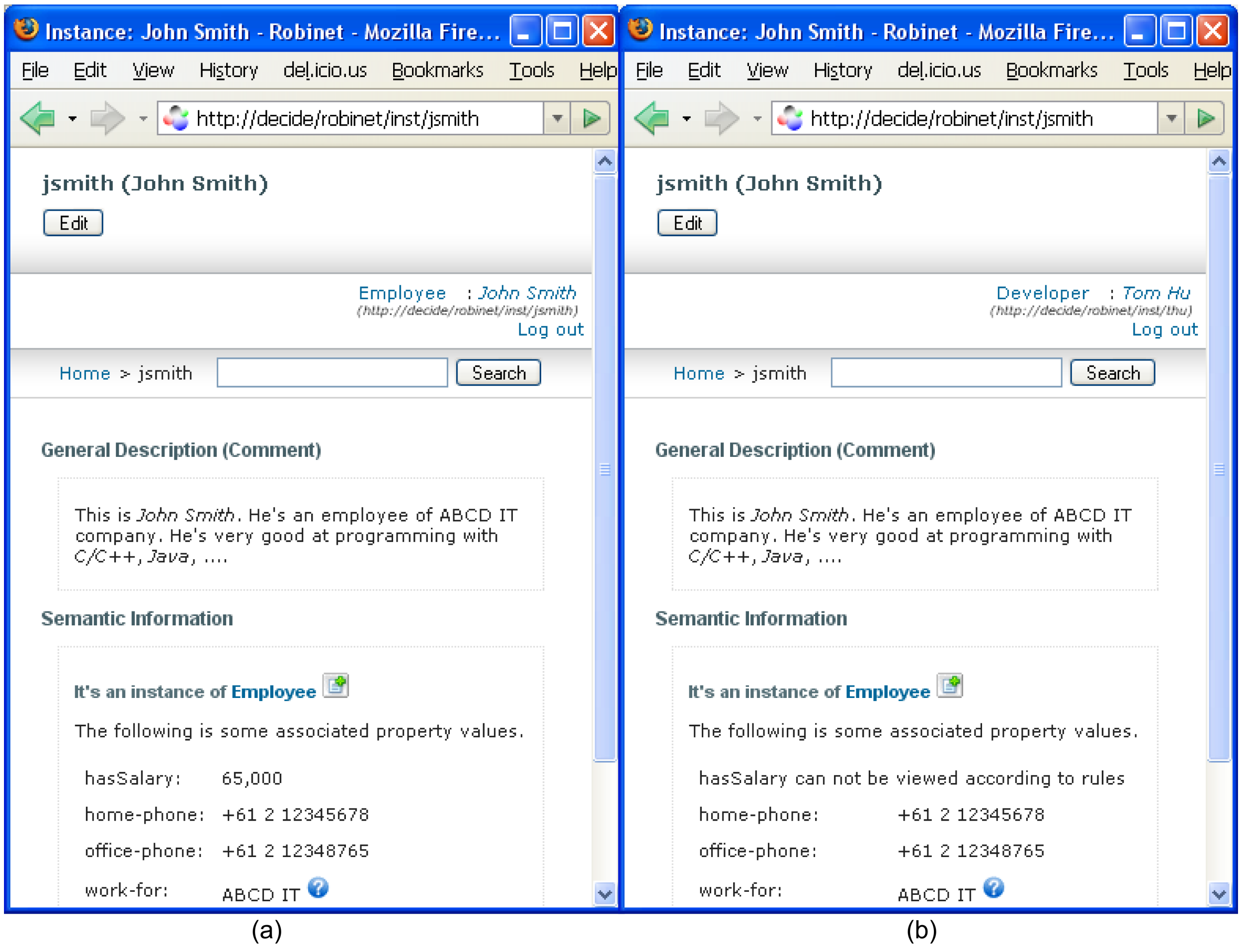
Obviously, it is very difficult to create a complete rule set to cover all operations properly, according to a given security requirement, if the domain is very complex. Sometimes, even the security requirement itself may not initially be very clear. In this case, we need to implement rule engineering to approach the ultimate security goal.

We can define a basic rule set with relatively low priority first and then add new rules with higher priority to override some old rules, as shown by Rules (9) and (10). A re-organization of these rules (both old and new) should also be allowed whenever it is necessary. Through this continuing process, we can finally meet the desired security requirement.

According to different situations, we can provide two approaches.

(1) For a situation when security is considered very seriously, we start with a set of very rigid rules and then adjust them thoughtfully by adding new permission rules or relaxing exiting rejection rules.

(2) In a very casual situation, we can start with rules that permit almost every operations. More rejection rules can be added in as more restrictions are needed.


Instance: John Smith - Robinet - Mozilla Fire...
File Edit View History del.icio.us Bookmarks Tools Help
http://decide/robinet/inst/jsmith
Edit
Log out
Home > jsmith
Search
(a)
(b)

(a)

jsmith (John Smith)

Employee : *John Smith*
(http://decide/robinet/inst/jsmith)

General Description (Comment)

This is *John Smith*. He's an employee of ABCD IT company. He's very good at programming with C/C++, Java, ...

Semantic Information

It's an instance of Employee

The following is some associated property values.

hasSalary:	65,000
home-phone:	+61 2 12345678
office-phone:	+61 2 12348765
work-for:	ABCD IT

(b)

jsmith (John Smith)

Developer : *Tom Hu*
(http://decide/robinet/inst/thu)

General Description (Comment)

This is *John Smith*. He's an employee of ABCD IT company. He's very good at programming with C/C++, Java, ...

Semantic Information

It's an instance of Employee

The following is some associated property values.

hasSalary can not be viewed according to rules

home-phone:	+61 2 12345678
office-phone:	+61 2 12348765
work-for:	ABCD IT

Figure 5: Different views of property values of the instance controlled by rules (a)`hasSalary` is viewable by `Employee` John himself; (b)`hasSalary` is not viewable by `Developer` Tom.

A Prolog interpreter "JLog"[4] is used to interpret and infer on the database of facts and clauses. After the rule set is tested and validated through rule engineering, access permissions can be consulted through the Prolog interpreter.

6. Further Discussion

The advantages and limitations of the proposed web ontology data management system in comparison with semantic wiki systems and conventional web based content management systems is discussed below.

The main advantage of the web ontology data management system over semantic wiki systems is the added flexible access control. As we have already indicated, reliability and security are problems of existing semantic wiki systems. Our system uses rules to semantically restrict operations on contents according to

[4]http://jlogic.sourceforge.net/

different participants. The enforcement of the rules may help protect certain information from unwanted editing. In addition, by editing rules, we can achieve different levels of security with greater flexibility.

Conventional web and database techniques can be used to implement a similar system to perform content management, which is a common solution nowadays. Compared to this solution, to develop a content management system for a certain domain using our approach, the major tasks include:

(1) to model the domain ontology schemas; and

(2) to define rules restricting operations on data.

Once these tasks are performed successfully, the system is almost ready. Less programming tasks are involved. In addition, while conventional content management systems require additional approaches to turn their managed contents into formats conforming to semantic web standards, our system directly manages semantic contents. Therefore, it is more compatible with the emerging semantic web.

We have tested the *robinet* system and asked a group of people to use and evaluate the system. An ontology schema describing "student", "staff", and so forth in the university context was used in the system for the evaluation. The overall result of the initial evaluation is satisfactory. It reveals that with a properly developed domain ontology schema, the system can efficiently help users to publish and manage web ontology data. In addition, by simply editing rules for access control, different access requirements can be flexibly met. We are currently planning more detailed and systematic evaluations to gather more feedback for the system. A test version is deployed at the address http://decide.it.uts.edu.au/robinet.

Some limitations to our system may exist. For our system to work properly, basic ontology schemas for the system need to be well developed so that they can reflect the domains appropriately, as the system currently does not allow users to edit the ontology schemas directly. This limitation is added to help simplify the modeling of access control because it becomes quite complicated to do this when ontology schemas themselves are constantly changing due to user modification. Further studies will be conducted to deal with the situation when ontology schemas are allowed to directly change. In addition, while the rule mechanism for access control can give us a lot of flexibility, it also implies that efforts need to be made to create a proper set of rules for the necessary security requirement. Initially, this study was motivated by certain scenarios mentioned earlier in the paper (e.g., IT company, university). The focus of the study is on access control in a limited or bounded collaborative environment. When placing the access control model into an open environment, the scalability of the model and the implementation may require further study.

7. Conclusions and Future Work

This paper discussed the collaborative management of web ontology data with a focus on the enforcement of a flexible access control mechanism. It analyzed the desirable functions that a semantic wiki-like system should possess to manage data. A system framework (on which the *robinet* system is based) was presented to illustrate the design of such a system and the implementation of the key functions was discussed. In particular, a web ontology data management model that enables a flexible access control mechanism for the system was proposed and details of its rule-based implementation were presented. Our initial tests and evaluation have shown the advantages of the proposed web ontology data management system: flexible access control and relative ease of use as a platform to manage web ontology data.

Our future work includes further improvement of the system interaction by using more advanced web techniques and tighter and better integration with other function modules, for example, the function for data quality control. In addition, the limitations of the current system, as discussed in the paper, will be dealt with in a future study. How to utilize the managed ontology data to provide certain services through web applications or agents will also be studied.

Acknowledgements

The authors sincerely thank all reviewers for their valuable comments and suggestions. The work presented in this paper was partially supported by Australian Research Council (ARC) under discovery grant PD0880739.

References

[1] S. Auer, Powl - a web based platform for collaborative semantic web development, in: Proceedings of the 1st Workshop Scripting for the Semantic Web (SFSW'05), 2005.
URL `http://www.semanticscripting.org/SFSW2005/papers/Auer-Powl.pdf`

[2] S. Auer, S. Dietzold, T. Riechert, Ontowiki - a tool for social, semantic collaboration, in: Proceedings of the 5th International Semantic Web Conference (ISWC 2006), vol. 4273 of Lecture Notes in Computer Science, Springer, 2006.

[3] D. Aumueller, S. Auer, Towards a semantic wiki experience desktop integration and interactivity in wiksar, in: Proceedings of the 1st Workshop on the Semantic Desktop in conjunction with the 4th International Semantic Web Conference, 2005.

[4] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider, The description logic handbook: theory, implementation, and applications, Cambridge University Press, New York, 2002.

[5] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Scientific American 284 (5) (2001) 34–43.

[6] D. Brickley, R. Guha, Rdf vocabulary description language 1.0: Rdf schema. w3c recommendation (2004).
URL `http://www.w3.org/tr/2004/rec-rdf-schema-20040210`

[7] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, in: Proceedings of the 7th International Conference on the World Wide Web, 1998.

[8] S. Castano, M. Fugini, G. Martella, P. Samarati, Database Security, Addison-Wesley, 1995.

[9] E. Damiani, S. D. C. di Vimercati, C. Fugazza, P. Samarati, Extending policy languages to the semantic web, in: Web Engineering (ICWE 2004), Lecture Notes in Computer Science, Springer, 2004.

[10] N. Damianou, N. Dulay, E. Lupu, M. Sloman, The ponder policy specification language, in: Proceedings of the International Workshop on Policies for Distributed Systems and Networks, 2001.

[11] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, J. Sachs, Swoogle: a search and metadata engine for the semantic web, in: Proceedings of the 13th ACM international conference on Information and Knowledge Management (CIKM '04), ACM Press, New York, NY, USA, 2004.

[12] M. F. Fernández, D. Florescu, A. Y. Levy, D. Suciu, Declarative specification of web sites with strudel, VLDB Journal 9 (1) (2000) 38–55.

[13] J. Fischer, Z. Gantner, M. Stritt, S. Rendle, L. Schmidt-Thieme, Ideas and improvements for semantic wikis, in: Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), 2006.

[14] P. Fraternali, Tools and approaches for developing data-intensive web applications: a survey, ACM Computing Surveys 31 (3) (1999) 227–263.

[15] P. Fraternali, P. Paolini, A conceptual model and a tool environment for developing more scalable, dynamic, and customizable web applications, in: EDBT '98: Proceedings of the 6th International Conference on Extending Database Technology, Springer-Verlag, London, UK, 1998.

[16] F. L. Gandon, N. M. Sadeh, Semantic web technologies to reconcile privacy and context awareness, Web Semantics: Science, Services and Agents on the World Wide Web 1 (3) (2004) 241–260.

[17] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubezy, H. Eriksson, N. F. Noy, S. W. Tu, The evolution of protege: an environment for knowledge-based systems development, International Journal of Human-Computer Studies 58 (1) (2003) 89–123.

[18] B. Grosof, I. Horrocks, R. Volz, S. Decker, Description logic programs: combining logic programs with description logic, in: Proceedings of the 12th International Conference on the World Wide Web (WWW2003), 2003.

[19] M. A. Harrison, M. L. Ruzzo, J. D. Ullman, Protection in operating systems, Communication of the ACM 19 (8) (1976) 461–471.

[20] J. Hendler, Agents and the semantic web, IEEE Intelligent Systems 16 (2001) 30–37.

[21] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, Swrl: A semantic web rule language combining owl and ruleml (May 2004).

[22] A. D. Iorio, V. Presutti, F. Vitali, Wikifactory: a web ontology-based application for creating domain-oriented wikis, in: Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), 2006.

[23] M. Jakob, H. Schwarz, F. Kaiser, B. Mitschang, Modeling and generating application logic for data-intensive web applications, in: ICWE '06: Proceedings of the 6th International Conference on Web Engineering, ACM Press, 2006.

[24] Y. Jin, S. Decker, G. Wiederhold, Ontowebber: Model-driven ontology-based web site management, in: Proceedings of the 1st International Semantic Web Working Symposium (SWWS'01), 2001.

[25] L. Kagal, Rei: A policy language for the me-centric project, Tech. rep., HP Labs (September 2002).
URL `http://www.hpl.hp.com/techreports/2002/HPL-2002-270.html`

[26] A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca-Grau, J. Hendler, Swoop: A 'web' ontology editing browser, Journal of Web Semantics 4 (2) (2006) 144–153.

[27] J. Kleinberg, Authoritative sources in a hyperlinked environment, in: Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.

[28] A. Maedche, S. Staab, Ontology learning for the semantic web, IEEE Intelligent Systems 16 (2) (2001) 72–79.

[29] F. Manola, E. Miller, Rdf primer. recommendation, w3c (2004).
URL `http://www.w3.org/tr/2004/rec-rdf-primer-20040210`

[30] D. L. McGuinness, F. van Harmelen, Owl web ontology language overview. w3c recommendation (2004).
URL `http://www.w3.org/tr/2004/rec-owl-features-20040210`

[31] G. Mecca, P. Atzeni, A. Masci, G. Sindoni, P. Merialdo, The araneus web-based management system, in: SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of data, ACM Press, New York, NY, USA, 1998.

[32] B. Motik, I. Horrocks, R. Rosati, U. Sattler, Can owl and logic programming live together happily ever after?, in: Proceedings of the 5th International Semantic Web Conference (ISWC 2006), 2006.

[33] S. Rizvi, A. Mendelzon, S. Sudarshan, P. Roy, Extending query rewriting techniques for fine-grained access control, in: SIGMOD '04: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, ACM Press, New York, NY, USA, 2004.

[34] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, Role-based access control models, Computer 29 (2) (1996) 38–47.

[35] A. Souzis, Building a semantic wiki, IEEE Intelligent Systems 20 (5) (2005) 87–91.

[36] S. Staab, A. Maedche, Ontology engineering beyond the modeling of concepts and relations, in: Proceedings of the ECAI'2000 Workshop on Application of Ontologies and Problem-Solving Methods, IOS Press, Amsterdam, 2000.

[37] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. S. D. Wenke, Ontoedit: collaborative ontology development for the semantic web, in: Proceedings of the 1st International Semantic Web Conference, Sardinia, Italy, 2002.

[38] R. Tazzoli, P. Castagna, S. E. Campanini, Towards a semantic wiki wiki web, in: Proceedings of the 3rd International Semantic Web Conference (ISWC2004), 2004.

[39] G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, A. Uszok, Semantic web languages for policy representation and reasoning: a comparison of kaos, rei, and ponder, in: Proceedings of the 2nd International Semantic Web Conference (ISWC 2003), Lecture Notes in Computer Science, Springer, 2003.

[40] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, J. Lott, Kaos policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement, in: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, 2003.

[41] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, R. Studer, Semantic wikipedia, in: Proceedings of the 15th International Conference on the World Wide Web (WWW'06), ACM Press, New York, NY, USA, 2006.

[42] C. Wang, J. Lu, G. Zhang, Integration of ontology data through learning instance matching, in: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006), 2006.

[43] C. Wang, J. Lu, G. Zhang, A constrained clustering approach to duplicate detection among relational data, in: Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007), 2007.

[44] C. Wang, J. Lu, G. Zhang, X. Zeng, Creating and managing ontology data on the web: a semantic wiki approach, in: Proceedings of the 8th International Conference on Web Information Systems Engineering, Nancy, France, 2007.