2011-01-01

A Web Site-Level Implementation Of Owl Sameas Predicate In Drupal

Patricia Esparza
University of Texas at El Paso, pesparza3@utep.edu

A WEB SITE-LEVEL IMPLEMENTATION OF OWL SAMEAS PREDICATE IN DRUPAL

PATRICIA ESPARZA

Department of Computer Science

APPROVED:

Paulo Pinheiro da Silva, Ph.D., Chair

Ann Q. Gates, Ph.D.

Ricardo von Borries, Ph.D.

Benjamin C. Flores, Ph.D.
Interim Dean of the Graduate School

A mi esposo Eduardo,

a mis papás Rosa y Andrés,

y a mis hermanos Andrés y Abraham.

Los amo.

A WEB SITE-LEVEL IMPLEMENTATION OF OWL SAMEAS PREDICATE IN DRUPAL

by

PATRICIA ESPARZA, B.S

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

December 2011

ACKNOWLEDGEMENT

First and foremost I offer my sincere gratitude to my advisor Dr. Paulo Pinheiro da Silva, my mentor at the Computer Science Department at The University of Texas at El Paso, who has supported me throughout my thesis with his enduring patience, effort, and advice. I am greatly indebted to him for his guidance through my Master's degree program; he was constantly driving me with energy and providing explanations when I felt lost. Thanks Dr. Pinheiro for giving me the opportunity to discover the research world.

Also, I want to express my most sincere gratitude to the members of my graduate committee, Dr. Ann Q. Gates, Vice President of Research of the Office of Research and Sponsored Projects at The University of Texas at El Paso, and Dr. Ricardo von Borries, Associate Professor at the Electrical and Computer Engineering Department at The University of Texas at El Paso. Their leadership and suggestions were an important key in this effort. Special thanks to Dr. Gates, her guidance, feedback, encouragement, support, and expertise were invaluable to the completion of this work.

There are no words to thank Dr. Rodrigo Romero, Assistant Director of the Cyber-ShARE Center of Excellence at The University of Texas at El Paso. He believed in me and filled me with the confidence. I am indebted to Dr. Romero because of his encouragement, helpful talks, guidance, and invaluable support.

Thanks a lot to Aida Gándara for her incalculable help, advice, encouragement, patience, and support. She never ceased in her belief in me, always giving me her time in spite of her own responsibilities.

Special thanks to all my Computer Science friends. Their stories, encouragement, motivation, advice, jokes, talks, and gatherings were an important key for this achievement.

I want to express my deep-felt thankfulness to my awesome husband, Eduardo Esparza, who has always been there for me, regardless of which path I decide to go. His love, encouragement, support, effort, and never-ending patience have made me accomplish everything I have aspired to.

I am heartily grateful to my mother and father for supporting me throughout all my life, without their love, motivation, thoughtful advice, and personal sacrifices I would not be what I am now. Thank you for being the great parents you are.

Finally, I offer my thanks and regards to all of those who supported me in any aspect during the accomplishment of this thesis. I appreciate the academic and personal support of you all.

ABSTRACT

The Ontology Web Language (OWL) *sameAs* predicate states that if the predicate holds between two Uniform Resource Identifiers (URIs) *A* and *B,* then an object identified by *A* is the *same* object identified by *B*; its purpose is to enable machines to understand that both objects are the same although they may have distinct Web identifiers, i.e., URIs. The proliferation of the use of this predicate has brought several issues to the Web, e.g., high cost of manually creating *sameAs* statements, inconsistent use of the predicate leading to distinct objects to be identified as being a single object, and too many *sameAs* statements prevent efficient reasoning with objects.

The goal of this thesis is to mitigate the proliferation of statements based on the *owl:sameAs* predicate by avoiding its use for every single pair of object identifiers that may be represented in multiple Web sites. In other words, instead of creating the same object *person* in multiple Web sites, the object of type person will be created at only one Web site, which is called *master server*, and a materialized reference of this object will be automatically created to support the reuse of the object in many other Web sites, called *receiver servers*, maintaining the object the same URI used for the object in the *master server*.

To show the effectiveness of the approach, this thesis describes a use case developed in the context of the collaboration between the NSF-funded Cyber-ShARE Center of Excellence at The University of Texas at El Paso (UTEP) and the TRUST Laboratory from the UTEP Computer Science Department. Both the Cyber-ShARE Center and the TRUST Laboratory have their own Web sites where they publish content such as their members' information, publications, and projects faculty is working on. The effort compared the number of commands to create users before and after the adoption of the *SameAs* implementation at the web-site level.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

A Web site is a collection of Web pages that disseminate information for commercial, educational, and personal purposes. Web sites can contain different types of information, e.g., members or users, products, contact information, and publications, which is often duplicated at other Web sites. The challenges with duplicated information are many, including issues of error propagation, inconsistency, and outdated information. The **goal** of this thesis is to define an approach that mitigates the proliferation of statements when identifying and reusing a single object across multiple Web sites. The approach will be based on the *owl:sameAs* predicate that states that an object identified by a URI *A* is the same object identified by a URI *B* different than *A* when *A* is an *owl:sameAs* predicate of *B.*

1.1 Problem Statement

The capability of cyber-environments to create multiple identifiers for any single object without the requirement of an information provider to reuse existing identifiers available on the Web is a key approach enabling information providers to easily publish content on the Web. However, the ability to create multiple identifiers for a single object is also a major source of information inconsistency. For instance, if a person is referenced on two Web sites and the work telephone numbers on these two home pages are different, the question arises as to which of these telephone numbers is correct. Is one of the Web sites old and contains an old work telephone number? Or is it the case that the person has indeed two valid work phone numbers. As we can see from this example, it is a challenge for Web content publishers to keep information consistent and up-to-date; as well as to avoid the propagation of errors across Web sites. It is difficult for a person with multiple Web pages to maintain his or her information

across the Web sites. Moreover, it is often the case that some Web sites may not be necessarily created and maintained by the person subject of the Web site. In this case, it may be impossible for the person to guarantee that his or her own information is consistent and up-to-date as published on the Web.

In addition to the error propagation, consistency, and outdated information issues, there is a need for machines to understand the following: how Web content represents real-world objects; what are the properties of these objects; and how to use the information about these objects and associated properties to perform useful tasks on behalf of humans. For example, if a machine needs to send an email to John Smith and invite him for a meeting, the machine may need to know all the following: how to find John Smith's home page; how to verify that John Smith is the person who needs to be invited; how to inspect the contact information in the home page for an email address; and how to use the email address to send John's invitation. A key issue with these tasks is that the machine may not know that the content of two given Web pages with the term "John Smith" is about a single person (or not). Chapter 2 presents the background work that has been done on semantic annotation to address this issue.

1.2 Challenges Associated with the *SameAs* Predicate

The *owl:sameAs* predicate has to be applied so a machine may understand that a given concept identified by a URI is the same concept identified by a distinct URI in another Web site. In term of reasoning, it may be expensive to maintain many *owl:sameAs* statements [18]. The reuse of URIs across multiple Web sites is an interesting and many times very desirable feature of the Ontology Web Language, but it may impose challenges to mitigate the proliferation of many *owl:sameAs* statements.

Privacy is another challenge related to the use of *sameAs* predicate. Web sites may contain information about resources such as site users, publications, and products. Some Web sites may contain metadata about their content. Data and metadata may be stored in databases with limited access to the general public because of privacy issues; there exists information that must be protected and not be publicly exposed.

Metadata on the Web often have information about real-world objects that may help machines understand the syntax of data. One of the true challenges regarding the reuse of data may be related to the understanding of their meaning. This thesis makes use of semantic Web technology, i.e., Resource Description Framework (RDF) [4] and Ontology Web Language (OWL) [3], to enable machine understanding the meaning of information and to be able to reuse it across Web sites.

1.3 Organization of Thesis

The thesis is organized as follows: Chapter 2 lays down a view of related work in the Semantic Web area, as well as provides a background about provenance, in particular the Proof Markup Language-Provenance (PMLP) ontology; Chapter 3 describes an example scenario to elucidate the problem being addressed. The *sameAs* mitigation approach is discussed in this chapter as well. Chapter 4 describes the evaluation process of the approach, and shows the results obtained. Chapter 5 provides an overview of the *sameAs* predicate; relevant issues about the use of this predicate are presented, a description of the solutions that have been developed, and a discussion of solutions that are similar to the work described in this thesis. Chapter 6 presents a summary of the thesis and future work.

Chapter 2

BACKGROUND

This chapter lays down a view of related work in the Semantic Web area, as well as provides a background about provenance, in particular the Proof Markup Language-Provenance (PMLP) ontology.

2.1 Uniform Resource Identifier (URI) and *sameAs* Predicate

A Uniform Resource Identifier (URI) is a string that identifies resources on the Web [1]. A resource is everything that can be identified [1], for example, a Web page, a document, an image, a person, an organization, and a country. Ontologies describe resources and relations among them. For instance, every resource that is on the Web may have at least one URI; and the content of this URI may describe specifics about the resource. It is important to mention that, on the Web, a URI identifies a resource uniquely and globally; that is, every URI is unique. Also, URIs are dereferenceable; that is, when they are found, they return data about the referent [22]. A content publisher is who defines what a URI identifies.

The Web Ontology Language (OWL) has a *sameAs* predicate, which is widely used in ontologies to annotate that two URIs identify the same resource [3]; in other words, the object identified by a URI *A* is the same object identified by a URI *B* if *A* is the same as *B*, and that the same concept can be identified by many other URIs. The *owl:sameAs* predicate needs to be applied so machines can understand that a given concept identified by a URI is the same concept identified by a distinct URI in another Web site. In term of reasoning, it may be expensive to maintain many *owl:sameAs* statements because now many additional properties may need to be aggregated for a single object every time new identifiers are identified to be from the common object [18]. The reuse of URIs across multiple Web sites provides a way to mitigate the

proliferation of many *owl:sameAs* statements, which is the focus of this thesis. The reuse of URIs is challenging among other reasons because information creators may have no infrastructure allowing them to discover resources on the Web.

2.2 Resource Description Framework (RDF)

Resource Description Framework (RDF) is "a general-purpose language for representing information in the Web" [4]. This language allows information publishers to document their knowledge about concepts and instances defined in ontologies, i.e., controlled vocabularies of concepts. Moreover, it enables the exposure of semantic information about Web content, allowing the reuse of it; and more importantly, allowing Web content to be searchable and retrievable by Semantic Web tools, i.e., RDF crawlers, search engines, and SPARQL engines [23].

Semantic Web technology is a standard way of representing real-world objects on the Web such that machines can understand and use the Web content. For example, RDF is a language used to encode Web information representing objects [2] and the Ontology Web Language (OWL) [3] specifies ontologies to enable reasoning by further limiting RDF data. For example, the FOAF ontology, acronym for Friend of a Friend, talks about concepts related to social networks such as people (viz., foaf:Person), organizations (viz., foaf:Organization), groups (viz., foaf:Group), and relationships among people and between people and organizations and groups. foaf:Person objects in RDF represent people objects.

Resources are RDF entities that represent objects such as people, documents, and organizations. For instance, a person named John may have two Web sites: one for his job and another for his personal activities. Person John is a resource, and this resource is identified by a URI on the Web. Since there are at least two Web sites with content about this specific John on

the Web, users may think that there are two John-person objects, but more importantly, people who read the two Web sites may say that these two John-persons may be the same person because they understand the content, and they may even know John. On the other hand, a machine cannot say if the contents of these two John-person Web sites are the same since the machine does not understand the content.

Now, this resource has two different URIs, but both are describing the same resource. With the previous example, it may be the case that two distinct URIs have information about a common object. This means that, without further analysis, a common object can be considered multiple objects. This is when the *owl:sameAs* predicate takes place; the predicate states that an object identified by a URI *A* is the same object identified by a URI *B* different than *A* when *A* is an *owl:sameAs* predicate of *B.* The *owl:sameAs* predicate needs to be applied if the machine is to understand that the objects identified by both *A* and *B* are the same. Or, in the case of this example, that the two pages with information about John are indeed information about a single person named John.

RDF triple stores store and retrieve metadata [7]. Querying capabilities of RDF triple stores can be leveraged since RDF triple stores are able to manage information about Web content, including their URIs. Then, assuming that Web servers are natively enhanced with RDF triple store capabilities and that Web servers can share RDF content, one can anticipate the reuse of RDF content in other servers more easily than without the RDF triple store capabilities.

2.3 Drupal Content Management System (CMS)

Drupal is an open source Content Management System (CMS) that allows users to build online communities; it simplifies the creation of Web sites because it provides managing aspects of site maintenance, i.e., access control, user and groups accounts, and the data storage in a

provided database [6]. Besides the fact that it is open source, some of the Drupal benefits include its flexibility, security, and online documentation, as well as the continual improvement of the product.

Drupal provides modules to add extra custom capabilities to Web sites. There exists a Drupal Development Initiatives which are active and focused on the creation and maintenance of several modules to provide solutions and functionalities for diverse Drupal servers' implementations [9]. This type of CMS maintains all its information in a database; so it is only available for specific number of users, depending on their privileges. This means that the information stored in Drupal is tied to it. In other words, since it is not available to everyone, it is hard to reuse this information across servers.

2.4 RDF Drupal Module

There are software solutions that expose semantic annotated information out of Drupal servers in RDF format. One of these solutions is a Drupal module called RDF Module [5]. The module allows Drupal Web site administrators to import any ontology or vocabulary, so that exposed information can be annotated with concepts coming from user-desired ontologies. Servers that host Web content need to conform with some specifications to be able to use this functionality, e.g., to a specific version of a PHP version. These kinds of issues make Drupal servers tight to specific module requirements, making this module difficult to use.

2.5 Provenance as Metadata

The purpose of provenance is to record every process execution step involved in producing a resource. In other words, provenance is to keep track of data in order to know how these data were created and modified. Proof Markup Language (PML) is a language that encodes provenance knowledge related to web resources [21]. The purpose of this language is to collect

evidence used to facilitate decisions about whether the information is trustable or not. That is, PML does not specify if information is trustable or not. It only keeps track of the information, i.e., who created the data, how it was collected, who has modified it, and how it has been processed. The scientist who will use the data will decide if he or she believes or trusts the data or not.

The Open Provenance Model (OPM) is a model for provenance; its purpose is to provide the exchange of provenance among systems, to provide the sharing and building of tools that support provenance, to define the provenance model in a defined mode, and to support the provenance representation [19]. That is, this model is focused on modeling provenance and its use and distribution.

To support the creation and use of provenance information, the W3C Provenance Working Group publishes W3C recommendations that define the language for exchanging provenance information among systems [26]. This group unifies the use of provenance for Semantic Web technologies, development, and promising standardization.

2.6 Proof Markup Language - Provenance (PMLP)

Ontologies represent knowledge. There exist several ontologies including the FOAF ontology, which describes persons, their information and relations to other people. One ontology of interest is the Provenance Markup Language-Provenance (PMLP) that describes information sources and their relations. Examples of information sources are organization, person, document, Web site, dataset, format, and publication. It describes the person who is a member of the Web site. This ontology models user information in a machine-readable way. More importantly, this information can be reused in various servers at almost no cost for users and servers

administrators. The PMLP ontology is one module of the Proof Markup Language (PML), which is a language that encodes dispersed provenance knowledge [21].

The connection between foaf:Person and pmlp:Person is that both of them describe people and their relations. This thesis leverages the PMLP ontology because it provides an integrated representation for information sources and justification on how web resources were derived or asserted.

Chapter 3

SAMEAS MITIGATION APPROACH

This chapter describes the *sameAs* mitigation approach and how it mitigates the proliferation of *owl:sameAs* predicates across multiple Web sites. The chapter presents a use case in which the approach is applied and it details the Drupal implementation of the approach.

3.1 Cyber-ShARE - TRUST Laboratory Use Case

The NSF-funded Cyber-ShARE Center of Excellence at The University of Texas at El Paso (UTEP) is a multidisciplinary Center with projects in Computer Science, Geology, and Environmental Science fields. The Center Web site (http://cybershare.utep.edu/) provides information about Center resources including associated people, projects, and project participation. Some of the information is provided by the Center Web site; however, this information often already exists in other affiliated departments and research groups Web sites.

For instance, the TRUST Laboratory, which includes participation with faculty from the Computer Science and Geology Departments, and the Environmental Science Systems Laboratory, which is associated with the Biology Department, have their own Web sites that contain information such as collaborations and project descriptions. Using this scenario as the use case, it can be observed that "redundant information" in the Center's Web site is often originally created and maintained independently of each department/research group Web site. Moreover, some of the information provided by the Center's Web site is available for internal consumption (private information) while the rest of the information is public.

These affiliated departments/research groups may have other projects that are unrelated to the Cyber-ShARE Center. For example, the TRUST Laboratory at UTEP collaborates with the Cyber-ShARE Center and conducts research on other Computer Science projects. Members of

the TRUST Laboratory collaborate with the Center. In this example, both Web sites are maintained independently. Members that are associated with the TRUST Web site and Cyber-ShARE Web site are entered independently; in principle, it may be incorrect to infer that information is always shared between the Center and its affiliated research groups; there is no mandatory relation even though they may contain the same information. Thus, both Web sites duplicate information that already exists at a collaborator's Web site.

In the use case, both organizations share instances of common concepts, e.g., sharing of *person* information. The concept *person* refers to people who are members of Cyber-ShARE and TRUST Laboratory; moreover, these *persons* are also Web site users that may have accounts for maintaining information on both Web sites, i.e., they may be information sources. Other common concepts among these Web sites are *publications*, *reports*, *funding*, *organizations*, and *documents*.

For the use case, two Drupal Web sites are taken into consideration:

- http://rio.cs.utep.edu/trust is called TRUST and plays the role of *master server*. In this Web, site users are created and maintained by Drupal administrators.
- http://rio.cs.utep.edu/cybershare is called Cyber-ShARE and plays the role of *receiver server*. In this Web site, TRUST users are currently created manually. Within the *sameAs* mitigation approach, users will be automatically created by the *sameas* module. The module is described on Section 3.3.

These two Web sites are about organizational units that share some members as collaborators, meaning that they share certain objects such as people. Some information of the TRUST Web site has to appear on the Cyber-ShARE Web site, e.g., information about the TRUST people who are also Cyber-ShARE members. In other words, in this particular use case,

it is desirable that every team member of TRUST, who is registered as a user at the TRUST Web site, should also be a registered user at the Cyber-ShARE Web site. More importantly, a user resource, whether it is Cyber-ShARE or TRUST, should have a unique URI, i.e., the URI will be from the Web site in which the user was originally created; in this use case, it will be according to the TRUST Web site.

In this use case, the flow of person information will always be from the TRUST Web site to Cyber-ShARE Web site. This means that once a user has been created for the TRUST Web site, the same user will be automatically created at the Cyber-ShARE Web site; but the user will be never created the other way, i.e., from the Cyber-ShARE Web site to the TRUST Web site. The information flow has one direction only: from the *master server* to the *receiver server*.

3.2 *SameAs* Vision for Drupal

To reuse information in Drupal, it is necessary to expose this information out of the CMS. Hence, we have developed a software solution to expose semantic provenance information encoded according the PMLP ontology that Drupal servers host. With this software solution, metadata stored in the database, including provenance, is exposed using RDF since it allows systematic, automatic integration of data from different sources. More importantly, CMS data is offered for reuse by other parties that are be tied to proprietary data storage or representation technology, e.g., a database dialect [4]. Once metadata is exposed out of Drupal servers, data may be searchable and retrievable by Semantic Web tools.

In the use case, a *receiver server* is defined as the server that receives and reuses information created, updated, and or deleted from another server, which is called *master server*. The *Master server* is responsible for creating content of a given type for the first time, as well as for generating the URIs that will identify created resources. Once the *master server* creates, updates,

or deletes content, information is then sent to the *receiver server(s)*, which reuse the data and metadata, and more importantly, reuse the URIs, which are the one provided by the *master server*. As a result of this information sharing strategy, the proliferation of statements based on *owl:sameAs* predicate is prevented.

3.3 Drupal *SameAs* Module

The software solution for the *sameAs* mitigation approach has been implemented as a Drupal module called *SameAs*. The implementation requires the following *reuse specification* defined between each pair of servers:

- The direction that the information will flow, e.g., from TRUST Web site (*master server*) to the Cyber-ShARE Web site (*receiver server*);
- The concept to be shared, e.g., *person*;
- The *receiver server* services URL;
- The *receiver server* domain;
- The *receiver server* API Key;

These specifications are defined at the *master server* in the *sameAs* administration section. The *receiver server* services URL, domain, and API key are necessary so both Web sites can communicate with each other.

There exist three operations related to users on Drupal servers: create, update, and delete resources, e.g., users of type person. These three functionalities provided by the *SameAs* module are described next.

3.3.1 Create a new user

The creation of a new user takes place at the *master server*, which is in charge of the creation of a URI that identifies to the new person resource. In Figure 1, it can be seen that a

new user named John wanted to be added to the *master server*, which is the box labeled as "TRUST Laboratory Web Site". Before the creation of user "John," it is necessary to determine if the user to be created does not exist at the *master server* and *receiver server*, which is the box label as "Cyber-ShARE Web site". Since user John exists already at the Cyber-ShARE Website, an account for this user is not created at any of the two Web sites. The error message that appears at the TRUST Laboratory Web site is displayed in the red box: "User not created. John user exists already at *receiver server*. Try another username". Otherwise, as shown in Figure 2, if user John does not exist at the TRUST Laboratory Web site and the Cyber-ShARE Web site, the *master server*, which is the box labeled as "TRUST Laboratory Web Site" will create the user and more importantly, a URI will be provided to identify it. The URI will be assigned by the TRUST Laboratory Web site, which is http://trust.utep.edu/pmlp/john.owl.

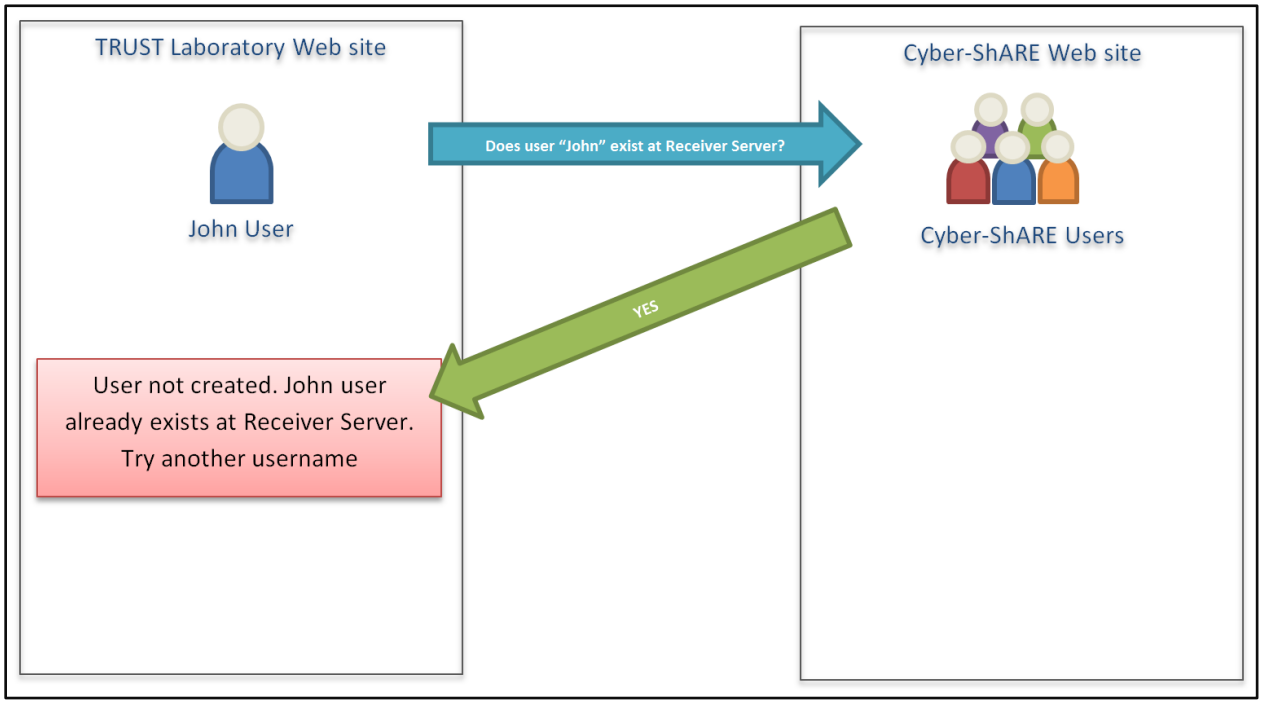

Figure 1. Creation of a user that already exists at the *receiver server*

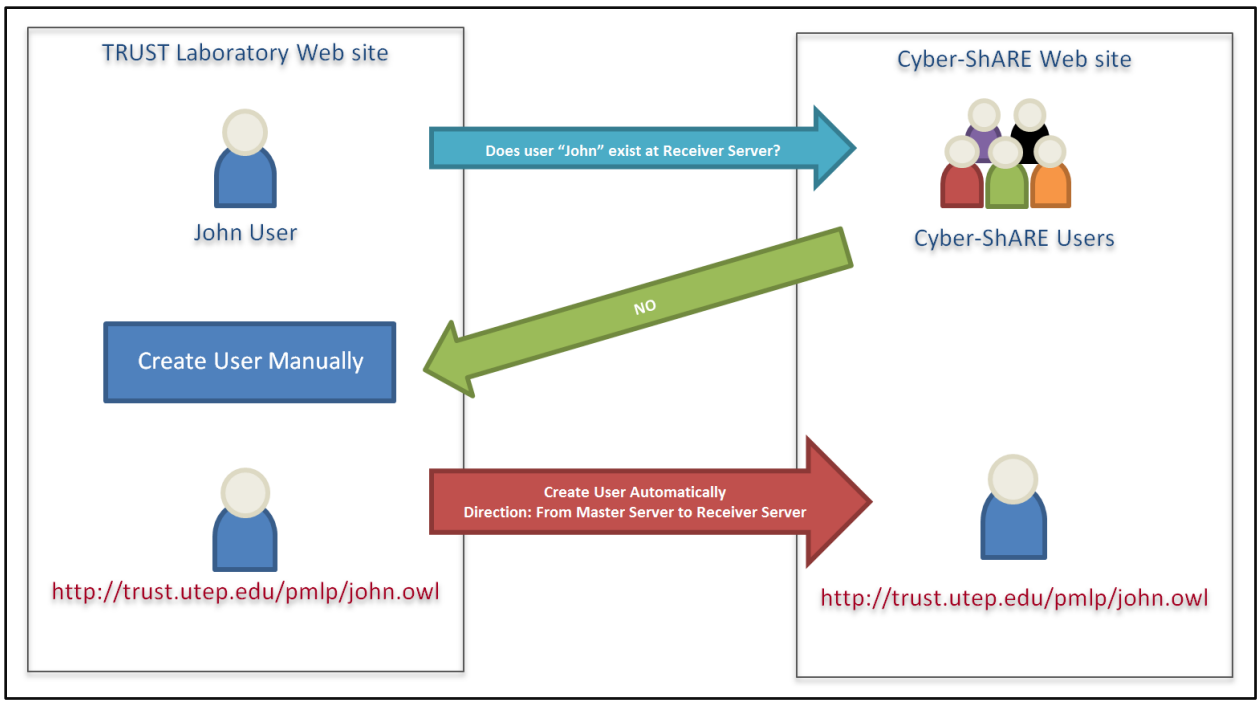

Figure 2. Creation of a new user that does not exists at the *master server* and *receiver server*

3.3.2 Update a user

This event takes place at the *master server* as well, which is the box labeled as "TRUST Laboratory Web Site". An important aspect is that the URI in the *receiver server*, which is the box labeled as "Cyber-ShARE Web Site" needs to remain the same as in the *master server*, no matter how many updates are made; the URI remains the same because it has been or could be already used by other ontologies. That is, there may be already ontologies that reference the URI generated by the TRUST Laboratory Web site to identify user John.

The update process, as shown in Figure 3, consists of checking if the resource to be updated, which is user John, exists at the *receiver server*, which is the box label as "Cyber-ShARE Web site." Since it is true, user John's information is updated on both Web sites. Since the URI will not change, it will be the same as the provided by the TRUST Laboratory Web site when the user was created.

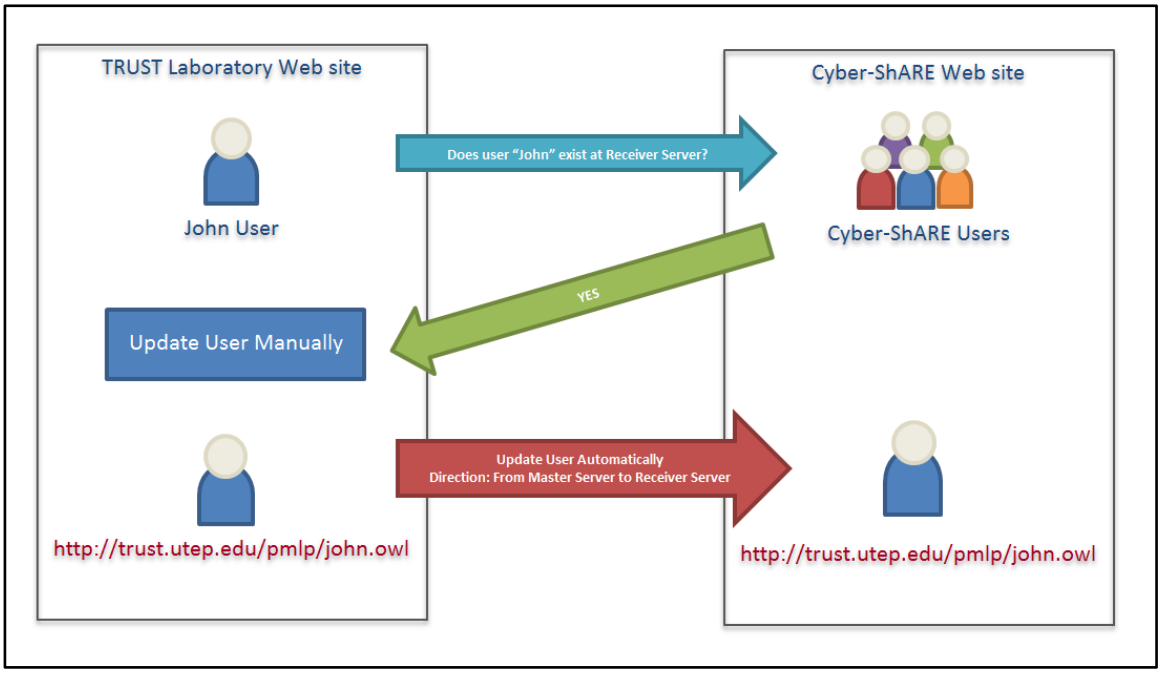

Figure 3. Update user process at the *master server* and *receiver server*

Figure 4 shows the contrary case, i.e., the situation in which John user exists only at the TRUST Laboratory Web site; hence user John is updated only at the *master server* and the URI remains the same as the provided by this Web site. As this figure shows, there are no changes at the Cyber-ShARE Web site.

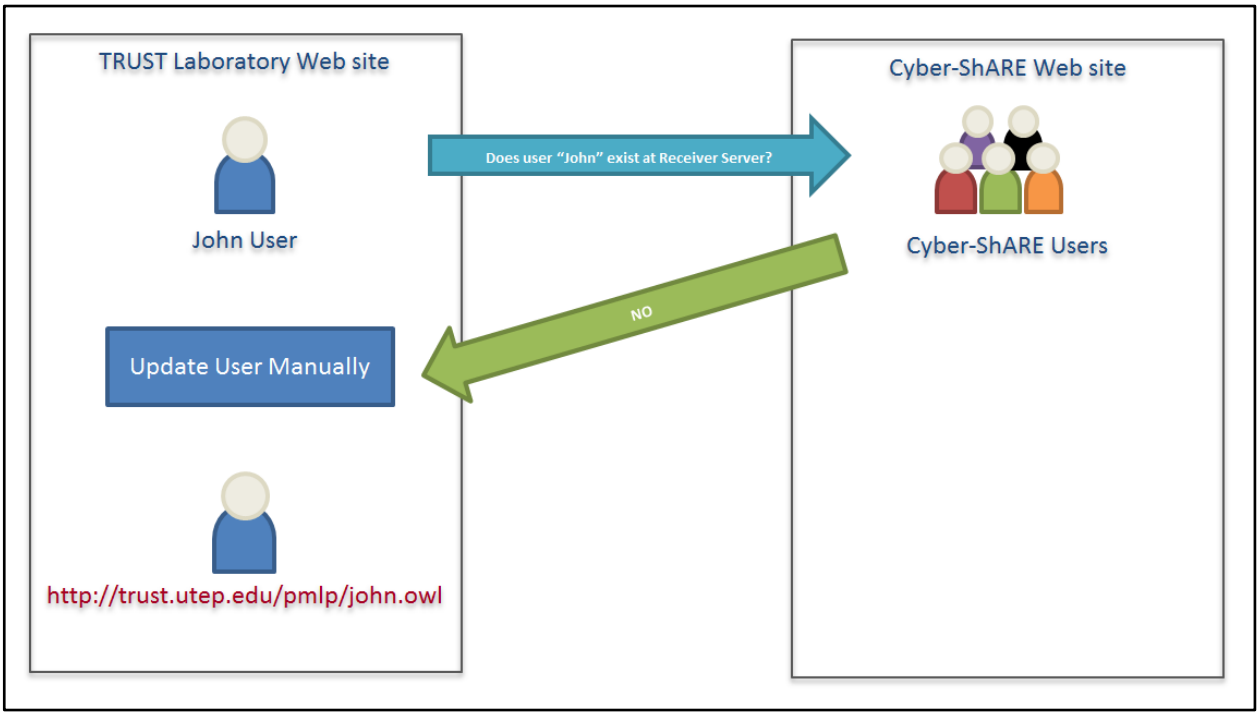

Figure 4. Update user process at the *master server*

3.3.3 Delete a user

The delete process is shown in Figure 5 and 6. As shown in Figure 5, the event takes place at the TRUST Laboratory Web site. It is determined if the user exists at the *receiver server*, which is the box labeled "Cyber-ShARE Web Site." Since it is true, the user will be updated as *blocked*, meaning that the user will no longer be active, but the user information will remain at the server. It is not possible to delete user John because, as in the update process, its URI will be deleted as well, and it may be the case that other ontologies refer to the URI of user John. This figure shows

how user John is updated to “blocked” at both Web sites. In addition, it shows that user John’s URI remains at both servers.

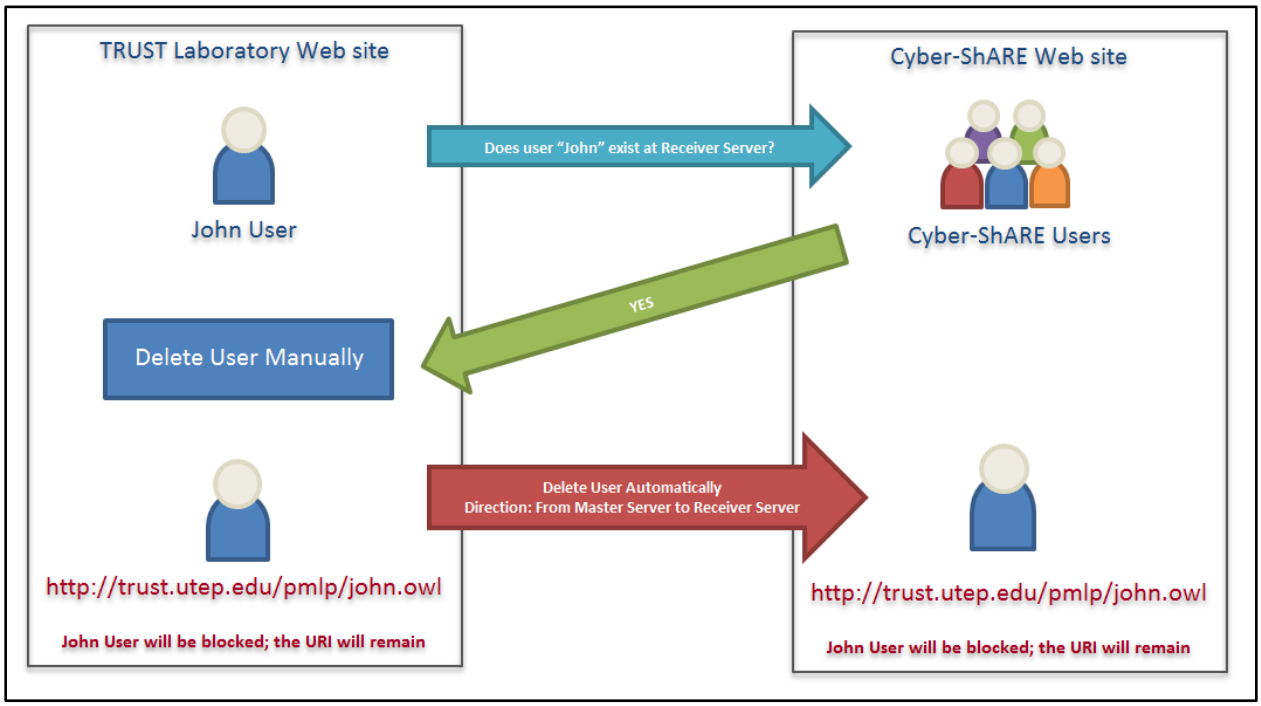

Figure 5. User deletion process at the *master server* and *receiver server*

Figure 6 shows the opposite case in which user John exists only at the TRUST Laboratory Web site; user John is only updated as *blocked* at this server, no changes are made in the Cyber-ShARE Web site.

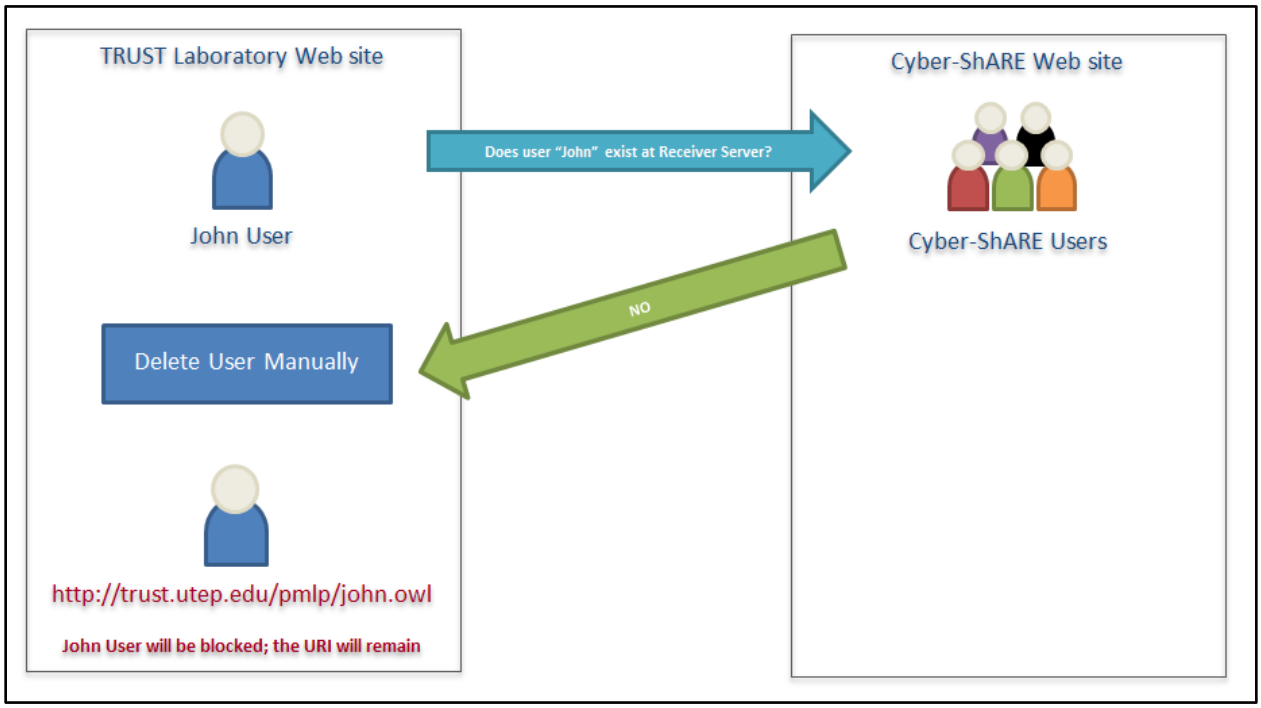

Figure 6. User deletion process at the *master server*

3.4 Drupal *PML* Module

URIs generated by the *master server* identifies the *person* resource, i.e., the user. To identify and describe each resource, the ontology PMLP is used with concept *person*. The metadata exposed by the PMLP ontology is shown in Figure 7.

One function of the *PML* module is to expose semantic information about users out of Drupal servers. The module constructs pmlp nodes for users that have an account on the system. Moreover, this module provides a view to display all the pmlp information that is on the server.

```
<rdf:RDF
    xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
    xmlns:pmlp='http://inference-web.org/2.0/pml-provenance.owl#'
    xmlns:owl='http://www.w3.org/2002/07/owl#'
    xmlns:xsd='http://www.w3.org/2001/XMLSchema#'
    xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'>
  <pmlp:Person rdf:about='http://rio.cs.utep.edu/ciservertest/pmlp/Peter-Smith.owl#Peter-Smith'>
     <pmlp:hasName rdf:datatype='http://www.w3.org/2001/XMLSchema#string'>Peter-Smith</pmlp:hasName>
  </pmlp:Person>
</rdf:RDF>
```

Figure 7. PMLP:Person information about users

Once the URI is created for a specific user, this URI will remain globally unique, consistent, persistent (even when the resource stops existing or becomes unavailable), and static. It needs to be that way because it might be the case that the created URI has been used by other ontologies.

3.5 Drupal *SameAs Service* Module

In every process at the *master server*, it is necessary to check if the user to be created, updated, and/or deleted exists at the *receiver server*. This module provides this functionality. Moreover, once a user is created, updated, and/or deleted at the *master server*, this service adds a new user at the *receiver server*, as well as updates the existing user's information. This service module uses the protocol XMLRPC. In order to authenticate at the *receiver server*, the *master server* needs to know the URL of the *receiver server*, the domain of the server, and the API key. These settings are specified at the *master server* in the *sameAs* administration section.

Chapter 4

EVALUATION OF APPROACH

4.1 Current Situation

People information on the TRUST Web site was added manually by Web site users with administrator's privileges; Cyber-ShARE Center people information, including TRUST members, is entered at the Center Web site manually by its own administrator. Both Web sites have their own Web site administrators. Information has been entered manually at both Web sites by different administrators and some person information is duplicated.

Currently, Cyber-ShARE Center Web site has twenty-eight users registered. TRUST Laboratory Web site contains eleven users registered. From these eleven TRUST Laboratory Web site users, five users have an account at the Cyber-ShARE Center as well. With these numbers, it was determined that some of the TRUST Laboratory members have an account at the Cyber-ShARE Center Web site as well; therefore, five users' information is being duplicated at both Web sites.

To enter a new user, the number of Drupal commands on the TRUST Laboratory Web site is eight. The same number of commands is applied when creating a new user at the Cyber-ShARE Web site. In order to register a person that collaborates with both TRUST Laboratory and Cyber-ShARE Center at both Web sites, the number of Drupal commands is sixteen; meaning that the process of creating a new user is being duplicated as well as the information. This means that in our use case, forty additional Drupal commands have been issued to create duplicate information that can be inconsistent and eventually become obsolete.

For example, a TRUST member called *John* has a URI at the TRUST Web site: http://trust.utep.edu/pmlp/john.owl and has a different URI at the Cyber-ShARE Web site:

http://cybershare.utep.edu/pmlp/john.owl. With this example we can say that every person has a different URI at every Web site, and that there is no level of URI reusability.

4.2 *SameAs* Mitigation Approach

With the use of the *SameAs* mitigation approach, the overall reuse of URIs increases across the TRUST Laboratory and Cyber-ShARE Web sites. The TRUST Laboratory Web site administrator has set up the *SameAs* module with specific information about the Cyber-ShARE Center Web site: server name, remote services URL, server domain, and API key. Then a user account is created for each TRUST Laboratory member at the TRUST Web site by the administrator. Then the same users will be created automatically at the Cyber-ShARE Web site by the *SameAs* module through the *SameAs Service* module.

With this approach every TRUST member registered as a user at the TRUST Web site will have a user account at the Cyber-ShARE Web site as well. The PMLP:Person metadata will be exposed at both servers by the *PML* module, more importantly, this URI will be unique and will be assigned by the *master server*, which in this case is the TRUST Laboratory Web site. In other words, every TRUST Laboratory member will have a TRUST Web site URI, which will be the same URI at the Cyber-ShARE Web site.

Using the previous example of the *John* TRUST member, his URI once he is registered at the TRUST Web site will be: http://trust.utep.edu/pmlp/john.owl. And *John* user will be registered automatically at the Cyber-ShARE Web site by the *SameAs* and *SameAs Service* module, but this time his URI will be the one provided by the TRUST Web site: http://trust.utep.edu/pmlp/john.owl. It is significant to mention that there exists a level of URI reusability. Moreover, the number of Drupal commands will decrease; this information will be shown in the next section.

The correctness of the implementation was made through testing it using functional test cases; Appendix *A* provides a description of them. The test cases were designed to evaluate the *PML* module, *SameAs* module, and *SameAs Service* module.

4.3 Results

The *SameAs*, *SameAs Service*, and *PML* modules were installed and enabled at both TRUST Laboratory and Cyber-ShARE Center Web sites. The *SameAs Service* module was set up with the Cyber-ShARE Web site information; this information includes server name, remote services URL, server domain, and API key. This information is necessary so the TRUST Laboratory Web site can communicate through the *SameAs* and *SameAs Service* module with the Cyber-ShARE Web site. The information needs to be entered once and can be updated any time.

Three users were created at the TRUST Laboratory Web site; eight commands were executed at this Web site in order to create one user. Then, the *PML* module created a PMLP:Person instance for each user, creating a unique URI and exposing this semantic information of each user as well. Since the three new users did not exist at the Cyber-ShARE Web site, the *SameAs* module communicated to the Cyber-ShARE Web site *SameAs Service* module and created each of these three users at this Web site. Figure 8 shows the PMLP information of one of the users created.

```
<rdf:RDF
    xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
    xmlns:pmlp='http://inference-web.org/2.0/pml-provenance.owl#'
    xmlns:owl='http://www.w3.org/2002/07/owl#'
    xmlns:xsd='http://www.w3.org/2001/XMLSchema#'
    xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'>
  <pmlp:Person rdf:about='http://rio.cs.utep.edu/trust/pmlp/John-Smith.owl#John-Smith'>
     <pmlp:hasName rdf:datatype='http://www.w3.org/2001/XMLSchema#string'>John-Smith</pmlp:hasName>
  </pmlp:Person>
</rdf:RDF>
```

Figure 8. PMLP:Person Semantic Knowledge generated by the *PML* module at the TRUST Laboratory Web site

As a result, to create a new user at both Web sites took eight commands instead of sixteen, as mentioned in the *Current Situation* section; reducing the number of executed commands by 50%. More importantly, the results show that the approach of mitigating the use of the *owl:sameAs* predicate has been achieved, since the URI for each created user is in fact the same at both Web sites. The URI corresponds to the Web site that created the user the first time, the *master server*.

The PMLP information generated by the *PML* module has already been crawled by a TRUST group triple store [25]. The triple store has twenty PMLP *person* instances generated by the *PML* module, and 1,564 *person* and *format* instances generated by the TRUST group tool called *derivA* [24]. These instances are hosted by a Drupal based TRUST group framework called *CI-Server*; it facilitates the sharing of data and provenance about scientific research [13]. Also, this Web site contains PMLP information about *people* that are not Web site registered users; this PMLP information is generated by *derivA*. The *derivA* tool allows the creation of provenance by encoding it in PML and uploading it to the *CI-Server*; hence the *CI-Server* contains PMLP information about the object *person*: some of it is created by the *PML* module for Web site registered users, and some of it is created by the *derivA* tool, allowing the generation of PML for people that are not registered users. With this information, it can be said that there are several ways to generate PMLP information about the object *person*, and that the *PML* module solves one aspect of it.

Chapter 5

RELATED WORK

5.1 Use of *owl:sameAs* Predicate

There has been an increment on the generation of RDF to support Semantic Web technologies; i.e., the Linking Open Data project is producing and exposing knowledge in RDF, and some of these statements contain *owl:sameAs* predicates [17].

5.2 Problems regarding *owl:sameAs* Predicate

There exists several issues regarding the *owl:sameAs* predicate. One of them is that the predicate is often applied incorrectly; for instance, there exists resources that are not the same and/or are not related, and their URIs are linked using the *owl:sameAs* predicate. In other words, the problem exists because there is no certainty that two or more URIs represent in fact the same resource [11].

Another problem is when related resources are not annotated with the *owl:sameAs* predicate [15]. That is, when two or more URIs in fact identify the same resource and they are not annotated using the *owl:sameAs* predicate. For example, a person named John Smith might have three resources with URIs, i.e., “John Smith”, “John-Smith”, and “Smith, J.”. A human being can infer that these three URIs may be about the same person, but a machine cannot; unless they are annotated using the *owl:sameAs* predicate.

There exists a problem called *Co-reference*; this concept defines that different URIs refer to the same resource. It tries to ensure that two distinct resources do not share the same URI, and to identify when two URIs refer to the same resource [10], [11], and [12].

One more problem with respect to the *owl:sameAs* predicate is its proliferation. In fact, this predicate tend to be used in the wrong way, viz., most of its use violate the logical semantics of the *owl:sameAs* [14] and [8].

5.3 *owl:sameAs* Predicate's solutions

One solution to the missing use of the *owl:sameAs* predicate, that is, when two or more URIs describe the same resource and they are not annotated by the *owl:sameAs* predicate, is to consolidate the URIs [15]. In other words, a solution is to identify which URIs refer to the same resource and to consolidate this entities using statistical information about the way the *owl:sameAs* predicate is applied.

With respect to the *Co-reference* problem, the creation of a URI management called Consistent Reference Service (CRS) is a solution [12]. This service manages *co-reference* between the URIs that are on the Semantic Web. The CRS groups together URIs for the same resource in different contexts for each data provider server. When equivalence is detected by CRS, the groups containing the URIs are merged together and a new group is created. In order for the CRS to manage the URIs, it should be installed in each server that provides RDF; which is difficult because not all RDF provider servers have a CRS.

In other to avoid the violation of the *owl:sameAs* predicate semantics, a proposed solution is the alternative identity links that rely on specific graphs [14]. Specifically, to use the *owl:sameAs* predicate in specific Web subgroups, not in all the Web of Data. For example, to specify that the URI *A* is the same as URI *B* only within graph *X*.

There has been proposed several solutions to the *owl:sameAs* predicate issues so far; some of them propose to use this predicate in sub groups of the Web of Data, other solutions propose to manage URIs at each data provider server installing a service. There is not an approach like

the one proposed in this thesis: to mitigate the proliferation of the *owl:sameAs* predicate by reusing a resource identified by URI assigned at a *master server* and reuse it across *receivers servers*.

5.4 Open ID

OpenID Foundation allows users to use an existing account to sign in to multiple Web sites, with not need of creating new accounts or passwords [20]. This approach is possible since OpenID relies on two parties: an OpenID Provider, which implements the OpenID protocol to authenticate users; and an OpenID Identifier, which is the master server that provides a URI for users to identify with the OpenID providers. The authentication process is as follows: users register in an OpenID identifier, which stores the user information. Then when users want to login at a specific Web site that provides OpenID login, users need to provide their OpenID URI. Then the OpenID provider takes the OpenID provider and redirects users over there to authenticate. Once users authenticate at the service provider server, users are redirected back to the service provider carrying a statement of successful login. This allows users to avoid typing different usernames and passwords at every Web site they are trying to login.

OpenID is being rapidly adopted on the Web. Two years ago, more than 500 million of users were using OpenID, and more than 48,000 Web sites were OpenID providers [16]. As of 2011, there are more than one billion OpenID users and over 50,000 OpenID providers [20].

Some of the advantages of using OpenID are that it is open source, anyone can create and use an OpenID, and administrators can add OpenID to their Web sites and become OpenID providers without the necessity of being approved by anyone. Moreover, a significant number of major companies provide OpenID registration, i.e., Google, Yahoo, AOL, Flickr, and myOpenID [20].

On the other hand, one of the disadvantages of OpenID is that, if the user information is stolen, other people will have access to all the servers to which the original user had access to. Moreover, OpenID does not expose semantic information about users.

OpenID is related to this work in the sense that one of its purposes is to reuse information across Web sites, so users do not have to create an account at each Web site they want to be a member of. OpenID does not use the *sameAs* mitigation approach and does not expose semantic information about users. Moreover, OpenID focuses only on users, while the approach of this thesis is broader.

Chapter 6

CONCLUSION

This thesis was motivated by the need for multiple RDF statements based on *owl:sameAs* predicate that have risen on the Web, especially in the presence of potential incomplete and inconsistent predicates. The proposed solution to mitigate the proliferation of the *owl:sameAs* predicate was to create Web-site level specifications that are responsible for a controlled replication of minimum provenance information that reuse resources' URIs. This means that instead of applying the predicate to specify that two URIs identify the same resource, the resource is created only once at a single server and then reused at many other servers through a network of Web-site level predicate-base replication strategy.

With the evaluation provided it has been shown that it is possible to expose semantic information from Drupal Web sites, including the exchange of semantic information across them. The results shown that there was an improvement in number of executed commands saved, the reduction was of 50%. Also, it has been shown that shared semantic information is up-to-date because it is reused instead of being re-created, and that the presented solution mitigates the proliferation of the *owl:sameAs* RDF predicate.

6.1 Broader Impacts

This thesis makes a *SameAs* projection at The University of Texas at El Paso (UTEP); it shows how the *SameAs* mitigation approach will help UTEP to reuse the employee information across different Web sites. This projection makes the following assumptions: each organizational unit wants its faculty and staff to have an organizational unit level Web page, each faculty and staff may have their own Web page, and the Office of Research and Sponsored Projects (ORSP) may want every faculty and staff to have a profile in the Expertise system. ORSP maintains a

database of faculty and staff expertise, but not every faculty and staff maintains their information in the system.

A report about the UTEP organizational units with their faculty and staff members was requested to the Center for Institutional Evaluation, Research and Planning (CIERP) of the university. The report included the faculty or staff name, job title, and department to which the faculty or staff belongs to. The report also included only UTEP primary faculty and staff that are funded by UTEP departmental accounts only. The university supplements the budget with other accounts and they are excluded of this report.

The report showed that there are 177 organizational units and 401 faculty and staff members at the university. From this population, 37.66% belongs to only one organizational unit, 50.87% belongs to two organizational units, 10.22% belongs to three organizational units, and 1.25% belongs to four organizational units. Figure 9 shows the chart that provides this data. With these results we can say that some faculty and staff work in more than one organizational unit. For instance, some faculty and staff information may be duplicated at more than one UTEP organizational unit Web site. Since there exists a URI for each faculty and staff Web page, we can say that 37.66% of URIs identifies one resource only, but 62.34% of URIs identify to two, three, or four resources that are the same resource.

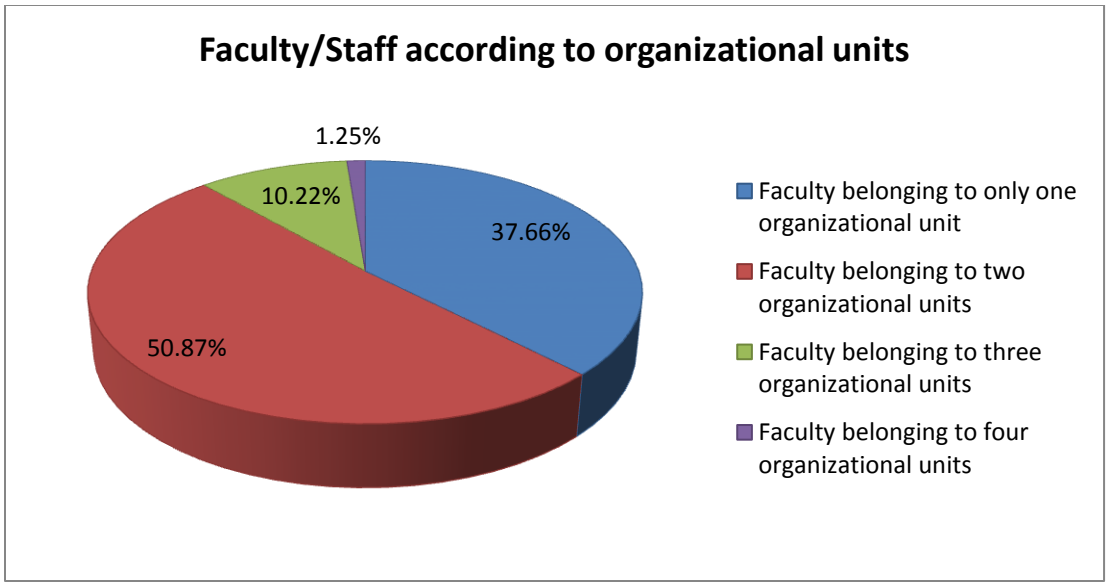

Figure 9. Faculty/Staff according to organizational units chart

According to the assumption that each organizational unit wants its faculty and staff to have an organizational unit level Web page and the CIERP report, there are 177 organizational units that may have Web sites that provide at least contact information about faculty and staff. Also, we can estimate that the level of duplicated information is 62.34% in the university; since this is the percentage that faculty and staff have duplicated information at two or more Web sites. A projection for the *SameAs* mitigation approach is that it will help to reuse Web content and URIs across the university Web sites by 62.34%. Another important projection is that UTEP Web content will be semantically annotated, allowing machines to understand the content and be able to use the resources information in order to perform useful duties for humans interests.

Considering the previous supposition and the assumptions that ORSP may want every faculty and staff to have a profile in the Expertise system and that each faculty and staff may have their own Web page; it can be said that faculty and staff information may be duplicated at three different Web sites. If it is taken into consideration that the number of Drupal commands to create a new user is eight, creating 401 faculty and staff profiles at one Web site will take 3,208

commands executed by the administrator. Moreover, executing the eight commands to create one user takes approximately three minutes, meaning that creating 401 profiles will take 20 hours or 2.5 working days for one Web site administrator.

Creating the same faculty and staff profiles at the organizational unit and ORSP Web sites will take 6,416 commands to be executed by a Web site administrator; this is one week of work. If every faculty and staff may want to have their own Web site, each faculty and staff would have to execute eight commands, bringing a total of 3,208 commands; this estimation assumes that each faculty and staff may have the elementary knowledge to create their Web sites by themselves. On the other hand, if an administrator would have to create faculty and staff profiles at three Web sites: organizational unit, ORSP and faculty and staff personal Web sites, it will take 9,624 commands to be executed by the administrator; meaning seven and a half working days. This assumptions and calculations show that it is expensive in terms of human work to create the same user information at three different Web sites manually. Moreover, these redundancies could lead to inconsistency of information because the manual process is error prone.

In terms of absolute number of duplicate URIs, there would be generated one URI for each faculty and staff profile, meaning that each single faculty and staff would have three different URIs that identify the same object. In other words, 1,203 URIs would be generated for 401 faculty and staff members. Using the *SameAs* approach there will be only 401 URIs, these URIs will be created at only one Web site and reused at the other two Web sites. With this approach, it will avoid the creation of 802 unnecessary URIs.

6.2 Future Work

Reusing different information of the source among servers is also future work. Moreover, it will be useful to query the reusable information through a RDF crawler so the proposed *SameAs* module can query for more information.

It will be useful to reuse certain objects, for example, a person may want to share her or his publications at one Web site and her or his contact information at a different Web site, but the person may not want to share both publications and contact information to both Web sites. An idea to approach it could be creating a mechanism so people may be able to select the PMLP classes they want to reuse across certain servers, e.g., PMLP:Person and PMLP:Publication.

6.2.1 Generalizing the *SameAs* Mitigation Approach

This sub section describes how the *SameAs* mitigation approach could be extended to implement other concepts of the PMLP ontology, as well as how the implementation could work among other types of content management systems.

Future work includes extending the current implementation of the *SameAs* mitigation approach so that other concepts could be applied, e.g., publications, organizations, and format. The approach infrastructure is already ready; the specifications do not need to change. A way to implement other concepts could be as follows: a new content type would be created at the *master server* and the *receiver server* for each concept that would be implemented. The *PML* module would need to create a content type for the concept to be implemented at the *master server*; the *SameAs* module would have to create the same content type through the *SameAs Service* module at the *receiver server*. Every content type for each concept would be created only once during the *PML* module installation process. The concept *publication* could be used as an example of how the process would be: Once a new publication was created at the *master server*, the *publication* would be created as a Drupal content, a URI that would identify the resource would be created as

well; then the same *publication* would be created at the *receiver server* by the *SameAs* module through the *SameAs Service* module; reusing the URI created by the *master server*. The *SameAs Service* module would create a *publication* content for the new *publication* at the *receiver server*.

The current implementation of the *SameAs* mitigation approach is applied to Drupal based Web sites only. An interesting idea is to implement this approach at different content management systems. In order to do it, it would be necessary to implement the specifications that the approach of this thesis describes in the programming language that supports the specific desired content management system. For example, if the approach wants to be applied in a CMS based on a C# programming language, the *SameAs* mitigation approach specifications would need to be implemented in a C# programming language.

REFERENCES

[1] BERNERS-LEE T., FIELDING R., AND MASINTER L. 2005. RFC 3986 - Uniform Resource Identifiers (URI): Generic Syntax, IETF. Available at http://www.isi.edu/in-notes/rfc3986.txt. January 2005.

[2] BECKETT, D., AND MCBRIDE, B. 2004. RDF/XML Syntax Specification (Revised), February 2004. W3C Recommendation.

[3] BECHHOFER, S., VAN HARMELEN, F., HENDLER, J., HORROCKS, I., MCGUINNESS, D., PATEL-SCHNEIDER, P., AND STEIN, L. A. 2004. OWL Web Ontology Language Reference. W3C Recommendation, February 2004.

[4] BRICKLEY, D. AND GUHA, R. (eds.), RDF vocabulary description language 1.0: RDF Schema, W3C Rec., http://www.w3.org/TR/rdf-schema/, Feb. 2004.

[5] CORLOSQUET, S., CYGANIAK, R., POLLERES, A., AND DECKER, S. 2009. RDFa in Drupal: Bringing Cheese to the Web of Data. In Proceedings of 5th Workshop on Scripting and Development for the Semantic Web at ESWC 2009, 2009.

[6] CORLOSQUET, S., DELBRU, R., CLARK, T., POLLERES, A., AND DECKER, S. 2009. Produce and Consume Linked Data with Drupal!. In ISWC 2009, vol. 5823 of LNCS, p. 763–778, Oct. 2009. Springer.

[7] DIETZOLD, S. AND AUER, S. 2006. Access Control on RDF Triple Stores from a Semantic Wiki Perspective. *In Proceedings of* Scripting for the Semantic Web Workshop at the ESWC, Jun 2006.

[8] DING, L., SHINAVIER, J., SHANGGUAN, Z., MCGUINNESS, D. 2010. SameAs Networks and Beyond: Analyzing Deployment Status and Implications of owl:sameAs in

Linked Data. In Proceedings of the 9th International Semantic Web Conference, ISWC. Shanghai, China. November 2010.

[9] Drupal Community. http://drupal.org/community

[10] JAFFRI, A., GLASER, H., AND MILLARD, I. 2008. Managing URI Synonymity to Enable Consistent Reference on the Semantic Web. In Proceedings of the Workshop on Identity, Reference, and the Web (IRSW) at ESWC2008, 2008. Tenerife, Spain.

[11] JAFFRI, A., GLASER, H., AND MILLARD, I. 2008. URI Disambiguation in the Context of Linked Data. In Proceedings of the 1st Workshop on Linked Data on the Web at WWW2008. Beijing, China.

[12] JAFFRI, A., GLASER, H., AND MILLARD, I. 2007. URI Identity Management for Semantic Web Data Integration and Linkage. In Proceedings of the Workshop on Scalable Semantic Web Systems. Vilamoura, Portugal.

[13] GANDARA, A. AND PINHEIRO DA SILVA, P. 2010. Provenance Support for Content Management Systems: A Drupal Example. The third International Provenance and Annotation Workshop, IPAW. New York, USA. June 2010.

[14] HALPIN, H. AND HAYES P. J. 2010. When owl:sameAs isn't the same: An analysis of identity links on the semantic Web. In Proceedings of the International Workshop on Linked Data on the Web, 2010. Raleigh, NC.

[15] HOGAN, A., POLLERES, A., UMBRICH, J., AND ZIMMERMANN, A. 2010. Some entities are more equal than others: statistical methods to consolidate Linked Data. In Proceedings of 4th International Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic (NeFoRS2010). Crete, Greece.

[16] JORSTAD, I., ANDERS, J., AND BAKKEN, E. 2009. Releasing the potential of OpenID & SIM. In Intelligence in Next Generation Networks, ICIN. 13th International Conference on, 2009, pp. 1-6.

[17] Linking Open Data Project http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[18] MCCUSKER, J.AND MCGUINNESS D. L. 2010. owl:sameAs considered harmful to provenance. In Proceedings of the ISCB Conference on Semantics in Healthcare and Life Sciences, 2010.

[19] MOREAU, L., PLALE, B., MILES, S., GOBLE, C., MISSIER, P., BARGA, R., SIMMHAN, Y., FUTRELLE, J., MCGRATH, R., MYERS, J., PAULSON, P., BOWERS, S., LUDAESCHER, B., KWASNIKOWSKA, N., VAN DEN BUSSCHE, J., ELLKVIST, T., FREIRE, J., AND GROTH, P. 2008. The Open Provenance Model. Technical report, Electronics and Computer Science, University of Southampton.

[20] OpenID Foundation. http://openid.net/

[21] PINHEIRO DA SILVA, P., MCGUINNESS, D. L., AND FIKES, R. 2006. A Proof Markup Language for Semantic Web Services. Information Systems, 31(4-5):381–395, June 2006.

[22] POLLERES, A., HOGAN, A., HARTH, A., AND DECKER, S. 2010. Can we ever catch up with the Web? Semantic Web -Interoperability, Usability, Applicability 1.

[23] Quick Intro to RDF. http://www.rdfabout.com/quickintro.xpd

[24] TRUST group software: derivA. http://trust.utep.edu/derivA/

[25] TRUST group Triple Store. http://trust.utep.edu:8080/sparql-pml/

[26] W3C Provenance Working Group. http://www.w3.org/2011/prov

APPENDIX A

General Information	
Test Case Name	PML Module - Configuration
Server Name	TRUST
Module/Version	PML - 6.x-1.0
Execution Date	11/7/2011
Executed by	Patricia Esparza
Description	Test the PML module - configuration section

Action	Expected Results	Pass/Fail
Install module: Go to Administer - Site Building - Modules and enable the pml module.	When installing the pml module, one table needs to be created in the database: - pml_visible_name To check the results, go to the MySQL database for the server that is being tested.	Pass
	When the pml module is enabled, three menus need to be created: - Provenance - PMLP - PMLJ To check the results, go to Administer - Site Building - Menus - Navigation and verify that the items are in the list of menus.	Pass
	URL aliases should be created for those three menus as follows: - provenance - provenance/pmlp - provenance/pmlj To check the results, go to Administer - Site Building - URL aliases and verify that the URLs are in the list.	Pass
Action	**Expected Results**	**Actual Results**
Uninstall module: Go to Administer - Site Building - Modules and disable the pml module. Click 'save configuration'. Then click on the 'Uninstall' tab, select the pml module and click on 'Uninstall'.	Once the pml module is uninstalled, one table needs to be deleted from the database: - pml_visible_name To check the results, go to the MySQL database for the server that is being tested.	Pass
Action	**Expected Results**	**Actual Results**
Disable module: Go to Administer - Site Building - Modules and disable the pml module.	When the pml module is disabled, three menus need to be deleted: - Provenance - PMLP - PMLJ To check the results, go to Administer - Site Building - Menus - Navigation and verify that the items are not in the list of menus.	Pass
	Three URL aliases should be deleted as follows: - provenance - provenance/pmlp - provenance/pmlj To check the results, go to Administer - Site Building - URL aliases and verify that the URLs are not in the list.	Pass

Table A1. PML Module: Configuration

General Information	
Test Case Name	PML Module - Menu Generation
Server Name	TRUST
Module/Version	PML - 6.x-1.0
Execution Date	11/7/2011
Executed by	Patricia Esparza
Description	Test the PML module - menus generation

Action	Expected Results	Pass/Fail
Enable PML module: Go to Administer - Site Building - Modules and enable the pml module.	When pml module is enabled, a menu should be displayed in the navigation menu: - Provenance	Pass
	Two submenus should be displayed under the Provenance menu: - PMLP - PMLJ	Pass
	Provenance menu should have the following URL: http://rio.cs.utep.edu/trust/provenance. URL depends on the server changes	Pass
	PMLP submenu should have the following URL: http://rio.cs.utep.edu/trust/provenance/pmlp. URL depends on the server	Pass
	PMLJ submenu should have the following URL: http://rio.cs.utep.edu/trust/provenance/pmlj. URL depends on the server	Pass

Table A2. PML Module: Menu Generation

General Information	
Test Case Name	PML Module - User Management
Server Name	TRUST
Module/Version	PML - 6.x-1.0
Execution Date	11/7/2011
Executed by	Patricia Esparza
Description	Test the PML module - user management

Action	Expected Results	Pass/Fail
Go to Administer - User Management - Users (depending on the server) and click on 'Add user'. Create a new user called 'John Smith'	When a new user is created, it should appear on the PMLP list, which is accessed through the PMLP submenu. To check the result, go to 'Provenance' and click the 'PMLP' submenu. "John Smith" should appear on the PMLP list.	Pass
	When a new user is created, a URI should be created to identify the user. To verify the result, click on the user 'John Smith' and check the URI. The URI should be http://rio.cs.utep.edu/trust/pmlp/John-Smith.owl#John-Smith	Pass
Go to Administer - User Management - Users (depending on the server) and click on 'Edit' under user 'John Smith'	When a user is edited the URI should remain the same. To verify the result, edit user "John Smith". The URI should be http://rio.cs.utep.edu/trust/pmlp/John-Smith.owl#John-Smith	Pass
Go to Administer - User Management - Users (depending on the server) and click on 'Edit' under user 'John Smith'. Then click the button 'Delete'	When a user is deleted, the user will be updated as "blocked". The URI will remain the same. The URI for user "John Smith" should be http://rio.cs.utep.edu/trust/pmlp/John-Smith.owl#John-Smith. To verify the result, click on the user 'John Smith' and check the URI.	Pass

Table A3. PML Module: User Management

General Information	
Test Case Name	SameAs Module - Administration Form
Server Name	TRUST
Module/Version	SameAs - 6.x-1.0
Execution Date	11/8/2011
Executed by	Patricia Esparza
Description	Test the SameAs module - administration form

Action	Expected Results	Pass/Fail
Go to Administer - Same As Admin. In the 'Known Servers' section do the following:		
Click REMOVE without selecting any content type	Error message should be displayed: "No content type was selected. Please click BACK and select one." A BACK button is displayed.	Pass
On no content type selected message, click BACK	Return to the main sameas module settings page. No message is displayed.	Pass
Select a content type managed by sameas and click EDIT	The settings for the selected server are displayed as: - Services URL - Server Domain - API Key Admin can edit them. Also, admin can click CANCEL or UPDATE	Pass
Click UPDATE	The sameas_known_servers table is updated for that specific server. Return to the main sameas module settings page. A successful message is displayed: "Server information updated successfully." To check the results, go to the MySQL database for the server that is being tested.	Pass
Click CANCEL	Return to the main sameas module settings page. No message is displayed. No changes made to the sameas_known_servers table. To check the results, go to the MySQL database for the server that is being tested.	Pass
Select a server managed by sameas and click REMOVE	The selected server is removed from the sameas_known_servers. A succesful message is displayed: "Server has been removed. Click BACK to go to the module settings page." Return to the main sameas module settings page. To check the results, go to the MySQL database for the server that is being tested.	Pass
Action	**Expected Results**	**Pass/Fail**
Go to Administer - Same As Admin. In the 'Add a known server' section do the following:		
Add a new server	After the administrator fills out the following fields: - Server Name - Remote Services URL - Server Domain - API Key and clicks "Add", the sameas_known_servers table is updated with the information entered. A successful message is displayed: "Server Added Successfully". To check the results, go to the MySQL database for the server that is being tested.	Pass
Add a server that already exists	Click "Add" without filling out any field. An error message should be displayed: "None of the above fields should be empty. Please provide the Server Name, Remote services URL, Server domain, and API key fields." No data should be added to the sameas_known_servers table.	Pass

Table A4. *SameAs* Module: Admin Form

General Information	
Test Case Name	SameAs Services Module
Server Name	TRUST
Module/Version	SameAs Services- 6.x-1.0
Execution Date	11/8/2011
Executed by	Patricia Esparza
Description	Test the SameAs Services module

Service Name	Action	Parameters	Expected Results	Pass/Fail
Service sameas.checkUser	Determine if user exists	name: John Smith	Since the user does not exist, the return array should be: Array ([exists] => 0)	Pass
	Determine if user exists	name: Patricia	Since the user exists, the return array should be: Array ([exists] => 1)	Pass
Service sameas.addUpdateUser	Create a new user	name: John Smith password: johnpassword mail: johnsmith@gmail.com access: 1 status: 1 is_person: 1 visible_name: John Smith creator: http://rio.cs.utep.edu/trust/pmlp/	The return object should be: stdClass Object ([uid] => 165 [name] => John Smith [pass] => 6f8f57715090da2632453988d9a1501b [mail] => johnsmith@gmail.com [mode] => 0 [sort] => 0 [threshold] => 0 [signature_format] => 0 [created] => 1323319129 [access] => 1 [status] => 1 [data] => a:0:{} [roles] => Array([2] => authenticated user) [profile_visible_name] => John Smith [profile_isperson] => 1 [profile_creator] => http://rio.cs.utep.edu/trust/pmlp/)	Pass
	Update an existing user	name: John Smith password: johnpassword **mail: jsmith@gmail.com** access: 1 status: 1 is_person: 1 visible_name: John Smith creator: http://rio.cs.utep.edu/trust/pmlp/	The return object should be: stdClass Object ([uid] => 165 [name] => John Smith [pass] => 6f8f57715090da2632453988d9a1501b [mail] => jsmith@gmail.com [mode] => 0 [sort] => 0 [threshold] => 0 [signature_format] => 0 [created] => 1323319129 [access] => 1 [login] => 0 [status] => 1 [data] => a:0:{} [roles] => Array([2] => authenticated user) [profile_visible_name] => John Smith [profile_isperson] => 1 [profile_creator] => http://rio.cs.utep.edu/trust/pmlp/)	Pass
	Delete an existing user	name: John Smith password: johnpassword mail: jsmith@gmail.com access: 1 **status: 0** is_person: 1 visible_name: John Smith creator: http://rio.cs.utep.edu/trust/pmlp/	The return object should be: stdClass Object ([uid] => 165 [name] => John Smith [pass] => 6f8f57715090da2632453988d9a1501b [mail] => jsmith@gmail.com [mode] => 0 [sort] => 0 [threshold] => 0 [signature_format] => 0 [created] => 1323319129 [access] => 1 [login] => 0 [status] => 0 [data] => a:0:{} [roles] => Array([2] => authenticated user) [profile_visible_name] => John Smith [profile_isperson] => 1 [profile_creator] => http://rio.cs.utep.edu/trust/pmlp/)	Pass

Table A5. *SameAs Service* Module

CURRICULUM VITA

Patricia Esparza was born on August 5, 1985. Patricia is the first daughter of Andres Reyes Villegas and Rosa Emma Hernández Guadian. She completed her Bachelor's in Computing Systems Engineering from Universidad Autónoma de Ciudad Juárez, Ciudad Juárez, México, in the fall of 2007. In the fall of 2008, she joined the Computer Science Graduate program at The University of Texas at El Paso (UTEP).

While pursuing a Master's of Science degree in Computer Science at UTEP, Patricia worked as a Teaching Assistant at UTEP, during the spring of 2009, and later worked as a Research Assistant at the same university from the summer of 2009 to the fall of 2010 at the Cyber-ShARE Center of Excellence under the supervision of Dr. Paulo Pinheiro da Silva. She was also a member of the TRUST Laboratory from the Computer Science department. During the spring of 2011 she worked as a Research Assistant for the same Center and for the Computing Alliance of Hispanic - Serving Institutions (CAHSI) under the supervision of Dr. Ann Q. Gates and Dr. Rodrigo Romero.

Patricia is currently working as a UTEP staff under the supervision of Dr. Ann Q. Gates; she collaborates in the CAHSI grant and the Innovation through Institutional Integration (I3) grant.

Permanent Address: 500 Talbot Ave. Spc. B3

Canutillo, Texas 79835