WebScripter: Grass-roots Ontology Alignment via End-User Report Creation

Baoshi Yan, Martin Frank, Pedro Szekely, Robert Neches, and Juan Lopez

Information Sciences Institute, University of Southern California
4676 Admiralty Way, Marina del Rey, California 90292
{baoshi,frank,szekely,rneches,juan}@isi.edu

Abstract. Ontologies define hierarchies of classes and attributes; they are meta-data: data about data. In the "traditional approach" to ontology engineering, experts add new data by carefully analyzing others' ontologies and fitting their new concepts into the existing hierarchy. In the emerging "Semantic Web approach", ordinary users may not look at anyone's ontology before creating theirs - instead, they may simply define a new local schema from scratch that addresses their immediate needs, without worrying if and how their data may some day integrate with others' data. This paper describes WebScripter, a tool for translating between the countless mini-ontologies that the "Semantic Web approach" yields. In our approach, ordinary users graphically align data from multiple sources in a simple spreadsheet-like view without having to know anything about ontologies. The resulting web of equivalency statements is then mined by WebScripter to help users find related ontologies and data, and to automatically align the related data with their own.

1 WebScripter Overview

WebScripter is a tool that enables ordinary users to easily and quickly assemble reports extracting and fusing information from multiple, heterogeneous Semantic Web sources in RDF Schema (RDFS) format[1]. Different Semantic Web sources may use different ontologies. WebScripter addresses this problem by (a) making it easy for individual users to graphically align the attributes of two separate externally defined concepts, and (b) making it easy to reuse others' alignment work. At a high level, the WebScripter concept is that users extract content from heterogeneous sources and paste that content into what looks like an ordinary spreadsheet. What users implicitly do in WebScripter (without expending extra effort) is to build up an articulation ontology containing equivalency statements. We believe that in the long run, this articulation ontology will be more valuable than the data the users obtained when they constructed the original report. The equivalency information reduces the amount of work future WebScripter users have to perform. Thus, in some sense, you do not just use the Semantic Web when you use WebScripter, you help build it as you go along.

[1] We will use RDFS for brevity in the remainder of the paperal though our tool and discussions equally apply to DAML(+OIL) and OWL.

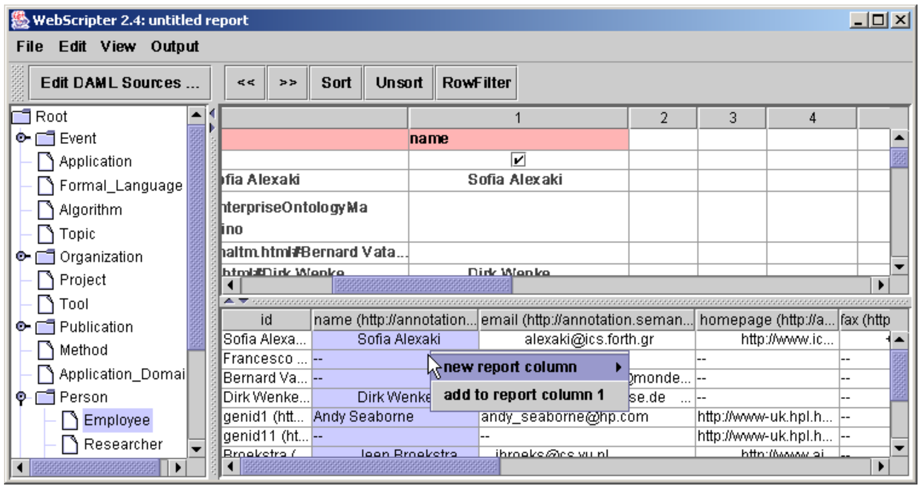
2 System Description

This section describes the current implementation of WebScripter by walking through a step-by-step example. In order to use WebScripter, users do not need to have knowledge of ontological languages. In this section we will describe how WebScripter help ordinary users locate RDFS sources, build a report and customize the representation of a report. We then show how the resulting ontology alignment data benefits other users in constructing similar reports by identifying related sources and aligning data.

2.1 Constructing a first report from scratch

Step 1: Load RDFS Data In this example our job is to maintain a list of researchers working on the Semantic Web. The first task is to find the URLs where the researchers put their data (which we presume to be in some RDF-based format for this example). Although locating RDFS sources is not WebScripter's focus, WebScripter provides some support for it by wrapping Teknowledge's Semantic Search Engine [1]. This search engine accepts queries in the format of triple patterns, and returns matches from the BBN's crawled ontology library [2]. Our wrapper helps users by transforming their keyword-based queries into triple patterns, submitting them to Teknowledge's Semantic Search Engine and extracting source URL's from the results. Later on we will discuss how WebScripter can help identify related RDFS sources in a collaborative filtering fashion. In this example, we will use two RDFS data sources, ISWC'2002 annotated author data [3] and ISI's Distributed Scalable Systems Division personnel data [4].

Step 2: Create a Report Figure 1 shows WebScripter just after loading the ISWC'2002 data. On the left side is a class hierarchy pane. Users can select a class to view its content in the lower right pane. The upper right pane is the report-authoring area. WebScripter offers three options for users to add a column to a report. (1) In the simplest case, users can select a column from a class and add it to the report, as shown in Figure 1. (2) Users can also type example data in the report-authoring area; WebScripter will then try to guess which column in which class the user is referring to. This is useful when users are lost in the class hierarchy. (3) In the most complicated case, users want to include information from different classes into a single report. We do not want to require users to understand the domain ontology in order to do that. For example, suppose users have already specified "name" and "email" for the instances of class "Person" in a report, and now they want to add information about the project a person works on, which is in the "Project" class. Instead of requiring users to specify how to go from the "Person" class to the "Project" class step by step, WebScripter will try to infer the ontological paths between these two classes, rank the paths first based on path length (shortest first) then by number of instance matches (more first), and lets users select (Figure 2). In our experience, the first entry listed

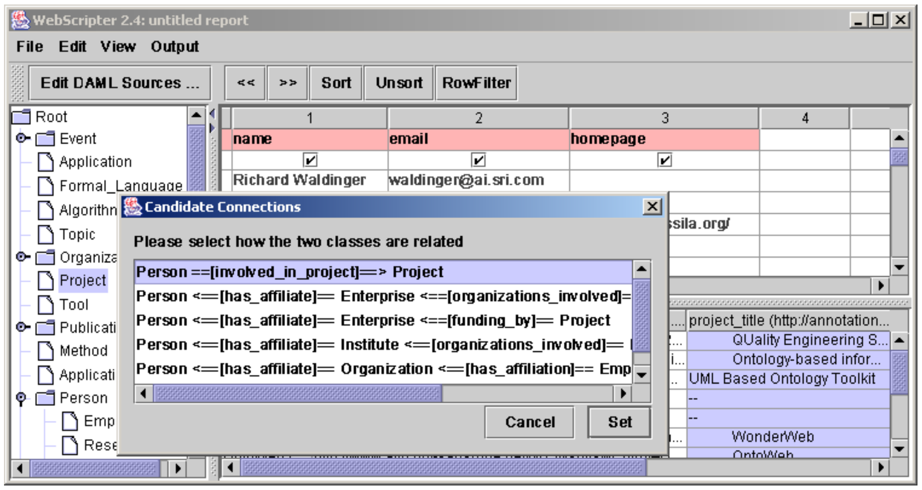


	1	2	3	4
	name			
	✓			
ofia Alexaki	Sofia Alexaki			
nterpriseOntologyMa ino				
haltm.html#Bernard Vata...				
html#Dirk Wenke	Dirk Wenke			

id	name (http://annotation...	email (http://annotation.seman...	homepage (http://a...	fax (http
Sofia Alexa...	Sofia Alexaki	alexaki@ics.forth.gr	http://www.ic...	
Francesco ...	--		--	--
Bernard Va...	--	monde...	--	--
Dirk Wenke...	Dirk Wenke	se.de	--	--
genid1 (htt...	Andy Seaborne	andy_seaborne@hp.com	http://www-uk.hpl.h...	--
genid11 (ht...	--	--	http://www-uk.hpl.h...	--
Broekstra (...	Jeen Broekstra	jbroeks@cs.vu.nl	http://www.ai...	--

Fig. 1. WebScripter GUI: The left pane shows the class hierarchy of ISWC'2002 data; the lower right pane shows all the instance data for the selected class. Users can add columns from this pane to their report in the upper right pane.

(the one with the shortest ontological path and which fills the most blanks in the report) is virtually always the desired choice.



	1	2	3	4
	name	email	homepage	
	✓	✓	✓	
	Richard Waldinger	waldinger@ai.sri.com		
			sila.org/	

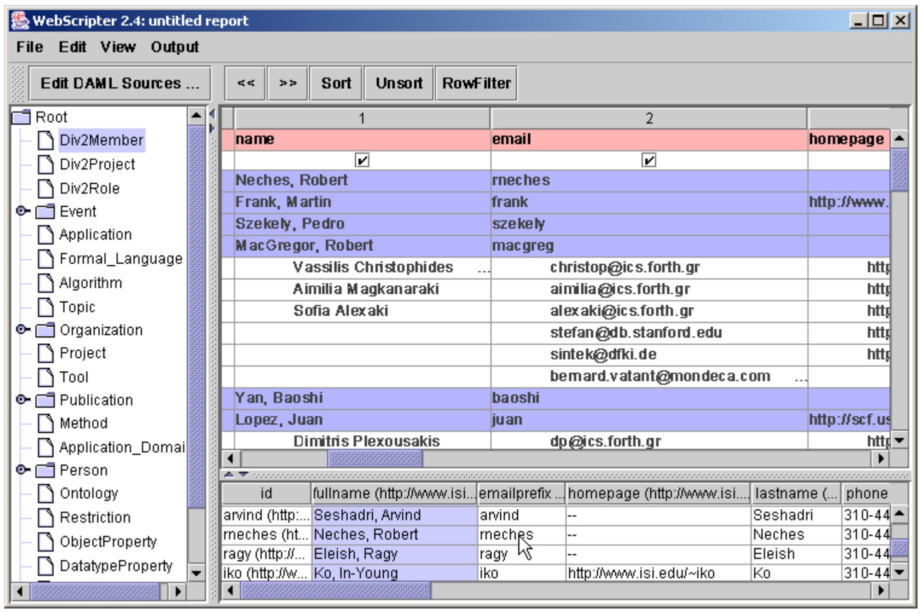
Please select how the two classes are related

Person ==[involved_in_project]==> Project
Person <==[has_affiliate]== Enterprise <==[organizations_involved]=
Person <==[has_affiliate]== Enterprise <==[funding_by]== Project
Person <==[has_affiliate]== Institute <==[organizations_involved]==
Person <==[has_affiliate]== Organization <==[has_affiliation]== Emp

project_title (http://annotation...
QUality Engineering S...
Ontology-based infor...
UML Based Ontology Toolkit
--
--
WonderWeb
OntoWeb

Fig. 2. Ontological Path Inference: When users add a column to the report that represents a new class, WebScripter detects the possible paths between these two classes and lets the user choose.

Step 3: Align data from multiple sources In our running example, the user is now done with adding ISWC'2002 author information to the report. Assume they happen to find ISI's researcher information via Teknowledge's Semantic Search Engine and want to include that in the report also. They basically repeat the previous steps of adding columns but this time they add the columns from ISI "Div2Member" class to the corresponding columns of the ISWC data (rather than adding it as new columns). Figure 3 shows the combined data from the two groups.



1	2	
name	email	homepage
✓	✓	
Neches, Robert	rneches	
Frank, Martin	frank	http://www.
Szekely, Pedro	szekely	
MacGregor, Robert	macgreg	
Vassilis Christophides ...	christop@ics.forth.gr	http
Aimilia Magkanaraki	aimilia@ics.forth.gr	http
Sofia Alexaki	alexaki@ics.forth.gr	http
	stefan@db.stanford.edu	http
	sintek@dfki.de	http
	bernard.vatant@mondeca.com ...	
Yan, Baoshi	baoshi	
Lopez, Juan	juan	http://scf.us
Dimitris Plexousakis	dp@ics.forth.gr	http

id	fullname (http://www.isi...	emailprefix ...	homepage (http://www.isi...	lastname (...	phone
arvind (http:...	Seshadri, Arvind	arvind	--	Seshadri	310-44
rneches (ht...	Neches, Robert	rneches	--	Neches	310-44
ragy (http://...	Eleish, Ragy	ragy	--	Eleish	310-44
iko (http://w...	Ko, In-Young	iko	http://www.isi.edu/~iko	Ko	310-44

Fig. 3. Aligning Data: In the upper right pane, the shaded data is from ISI, the light data from ISWC.

When users compose a report by putting together information from heterogeneous sources, there is some implicit and valuable information that can be inferred. First, by composing a report, users imply a (weak) association between sources, i.e., "one user who use this source also used that one", somewhat analogous to Amazon's book recommendations ("customers who bought this book also bought that one"). This association can help future users locate relevant RDFS sources. Second and more interestingly, by putting heterogeneous information together, users also imply a (similarly weak) equivalency between concepts from different ontologies. For example, from the report in Figure 3 WebScripter could infer that ISI's "Div2Member" class is equivalent to ISWC's "Person" class, ISI's "fullname" property is equivalent to ISWC's "name" property, and so on.

Table 1 shows the equivalency information inferred from the report in DAML format.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:wseq="http://www.isi.edu/WebScripter/2002/06/equivalencies#"
>
<rdfs:Class rdf:about="http://annotation.semanticweb.org/iswc/iswc.daml#Person">
  <daml:sameClassAs rdf:resource=
        "http://www.isi.edu/webscripter/div2-org.o.daml#Div2Member"/>
</rdfs:Class>
<rdfs:Property rdf:about="http://annotation.semanticweb.org/iswc/iswc.daml#name">
  <daml:samePropertyAs rdf:resource=
        "http://www.isi.edu/webscripter/div2-org.o.daml#fullname"/>
</rdfs:Property>
<rdfs:Property rdf:about="http://annotation.semanticweb.org/iswc/iswc.daml#email">
  <daml:samePropertyAs rdf:resource=
        "http://www.isi.edu/webscripter/div2-org.o.daml#emailprefix"/>
</rdfs:Property>
<rdfs:Property rdf:about="http://annotation.semanticweb.org/iswc/iswc.daml#homepage">
  <daml:samePropertyAs rdf:resource=
        "http://www.isi.edu/webscripter/div2-org.o.daml#homepage"/>
</rdfs:Property>
<rdfs:Property rdf:about=
        "http://annotation.semanticweb.org/iswc/iswc.daml#involved_in_project">
  <daml:samePropertyAs rdf:resource=
        "http://www.isi.edu/webscripter/div2-org.o.daml#workson"/>
</rdfs:Property>
<rdfs:Property rdf:about="http://annotation.semanticweb.org/iswc/iswc.daml#project_title">
  <daml:samePropertyAs rdf:resource=
        "http://www.w3.org/2000/01/rdf-schema#label"/>
</rdfs:Property>
</rdf:RDF>
```

Table 1. Resulting alignment axioms.

The alignment axioms shown above are the simplest ones, a direct alignment between two named classes or properties. Since WebScripter also supports joins (between two classes) and filtering (of instances), the alignment axioms can also be more complex. For example, if users want to build a report of just the ISI students, users need to add "Div2Member" instances to the report, do a join to their roles ("Div2Role") and filter the roles by "Student". The resulting equivalency is visualized as in Figure 4.

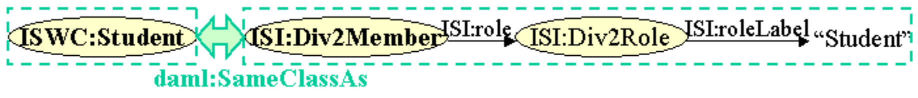


Fig. 4. Constructed-Class Alignment: An equivalency between a named class and a class defined by a join and subsequent fiter operation.

Figure 5 shows an axiom that defines the equivalency between two property sequences. This type of axiom can be captured with WebScripter (but we do not yet make use of it for our own alignment suggestions). To obtain the project name for a person, in the first case users simply follow the link "foo:projectName"; in the second case users need to follow the link "ISWC:involved_in_project", then the link "ISWC:project_title". Such "role chaining" is not expressible in current DAML or OWL. We are looking at RuleML [5] as an alternative.

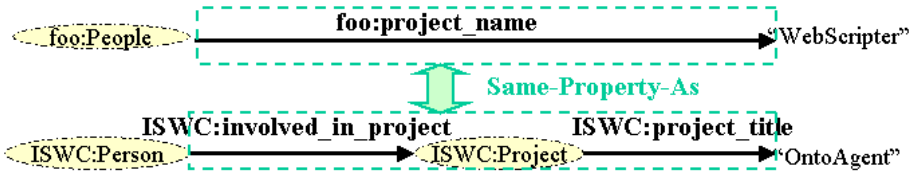


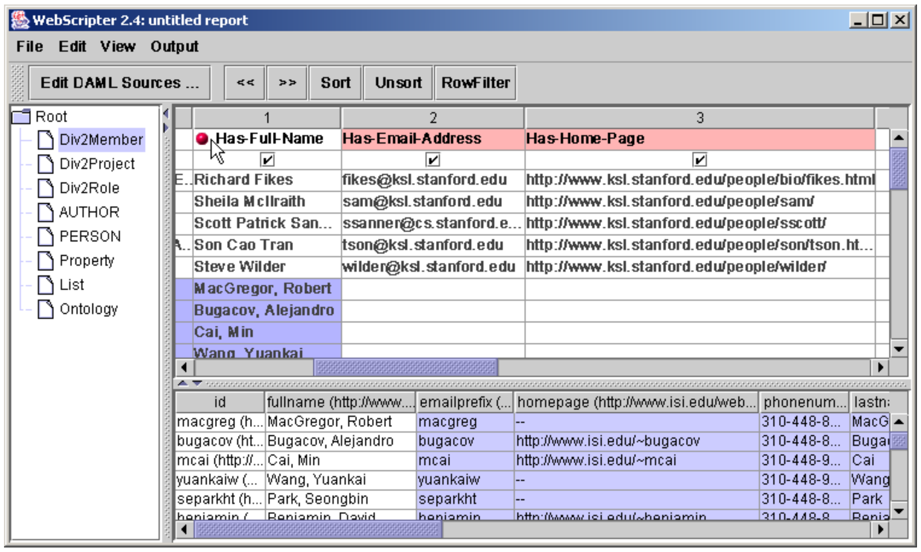
Fig. 5. Constructed-Property Alignment: An equivalency between two property sequences.

Current semi-automatic ontology mapping tools (see [26] for a good survey) are good at one-to-one element mapping and tend to deal less with alignment axioms as complex as shown in Figures 4 and 5, which WebScripter in some sense captures "for free" by providing an easy way for users to perform join and filtering during report authoring.

2.2 Constructing a report with automatic alignment support

The alignment axioms can be automatically published on a Web site and registered as a new DAML content root in BBN's DAML content library. Consequently, it can be used by Teknowledge's DAML search engine to extend user queries, and we expect it would benefit other applications as well. We also use the WebScripter-generated alignment axioms in WebScripter itself. WebScripter reads its default alignment axioms from a fixed location on our Web site to which anyone can contribute via WebScripter's "Easy Publish" menu. In this section we will quickly walk through an example of how the alignment axioms help users in report authoring.

In a nutshell, if users add any class or attribute to their report for which there are known equivalencies, WebScripter will mark them with a red "light bulb". If users click on the light bulb they can choose to import equivalencies. Accepting equivalencies can also automatically load known sources of the data, and then automatically aligns the attributes of the discovered classes. In this example, a user has constructed a new report which contains Stanford personnel data (the light area of the upper-right pane in Figure 6). She has just now manually aligned the Stanford and ISI names in the same column, and WebScripter responded by putting a lightbulb next to the the column name. This is an indication that WebScripter has equivalency information about the data she just added.



	1	2	3
	● Has-Full-Name	Has-Email-Address	Has-Home-Page
	✓	✓	✓
E..	Richard Fikes	fikes@ksl.stanford.edu	http://www.ksl.stanford.edu/people/bio/fikes.html
	Sheila McIlraith	sam@ksl.stanford.edu	http://www.ksl.stanford.edu/people/sam/
	Scott Patrick San...	ssanner@cs.stanford.e...	http://www.ksl.stanford.edu/people/sscott/
A..	Son Cao Tran	tson@ksl.stanford.edu	http://www.ksl.stanford.edu/people/son/tson.ht...
	Steve Wilder	wilder@ksl.stanford.edu	http://www.ksl.stanford.edu/people/wilder/
	MacGregor, Robert		
	Bugacov, Alejandro		
	Cai, Min		
	Wang, Yuankai		

id	fullname (http://www....	emailprefix (...	homepage (http://www.isi.edu/web...	phonenum...	lastn:
macgreg (h...	MacGregor, Robert	macgreg	--	310-448-8...	MacG
bugacov (ht...	Bugacov, Alejandro	bugacov	http://www.isi.edu/~bugacov	310-448-8...	Bugac
mcai (http://...	Cai, Min	mcai	http://www.isi.edu/~mcai	310-448-9...	Cai
yuankaiw (...	Wang, Yuankai	yuankaiw	--	310-448-9...	Wang
separkht (h...	Park, Seongbin	separkht	--	310-448-8...	Park
benjamin (...	Benjamin, David	benjamin	http://www.isi.edu/~benjamin	310-448-8	Benja

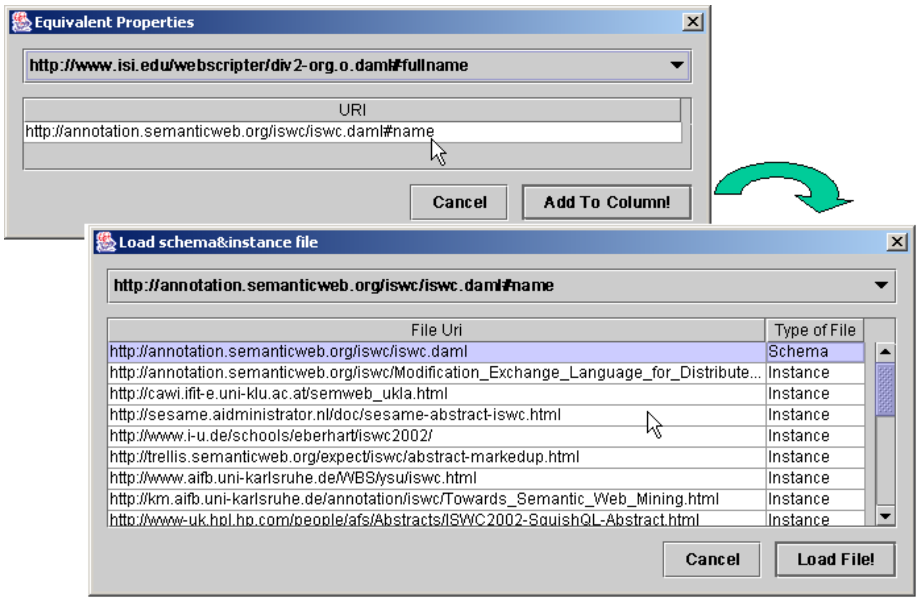
Fig. 6. Light bulbs tell the user that WebScripter can auto-align data for them.

When the user clicks on the light bulb, WebScripter lists the equivalence or equivalencies and asks the user to accept or reject them (upper window of Figure 7). In this example, WebScripter knows where ISWC people data is located, and also knows how the ISWC aligns with the ISI data; it knows that because they were aligned by someone else before (in Figure 3) and because that user chose to share the alignment information.

If the user accepts an equivalency, WebScripter displays a further dialog box that lets users decide which of the known source files of the equivalent classes/properties they could add to their report (lower window Figure 7). None, some, or all of the rows can be selected with the mouse, clicking the Load File button then adds the selected URLs to the data sources that are imported by the WebScripter report. In this example, if the user accepts the ISI/ISWC equivalencies and completes the Standord/ISI equivalencies and published those, WebScripter can now translate between all three ontologies.

2.3 Customizing WebScripter reports

WebScripter reports can be published in various formats including HTML and plain text. WebScripter first generates an XML representation of the report. Various XSL stylesheets are part of the WebScripter distribution, and you can define your own variation to customize the presentation. We are also currently working on a new version that will allow accessing WebScripter reports on mobile devices such as PDAs by intelligently reformatting the HTML to smaller screen sizes [20].



http://www.isi.edu/webscripter/div2-org.o.daml#fullname

URI
http://annotation.semanticweb.org/iswc/iswc.daml#name

http://annotation.semanticweb.org/iswc/iswc.daml#name

File Uri	Type of File
http://annotation.semanticweb.org/iswc/iswc.daml	Schema
http://annotation.semanticweb.org/iswc/Modification_Exchange_Language_for_Distribute...	Instance
http://cawi.ifit-e.uni-klu.ac.at/semweb_ukla.html	Instance
http://sesame.aidministrator.nl/doc/sesame-abstract-iswc.html	Instance
http://www.i-u.de/schools/eberhart/iswc2002/	Instance
http://trellis.semanticweb.org/expect/iswc/abstract-markedup.html	Instance
http://www.aifb.uni-karlsruhe.de/WBS/ysu/iswc.html	Instance
http://km.aifb.uni-karlsruhe.de/annotation/iswc/Towards_Semantic_Web_Mining.html	Instance
http://www-uk.hpl.hp.com/people/afs/Abstracts/ISWC2002-SquishQL-Abstract.html	Instance

Fig. 7. The upper window lets users accept or reject an equivalency per se. The lower window lets users decide if they also want to add data sources to their reports that contain aligned data.

3 Applications

WebScripter has turned out to be a valuable practical tool for the simple single-ontology case where there is only one schema but the instance data is distributed over many Web pages. For example, the Distributed Scalable Systems Division at ISI automatically pulls together its people page from many different DAMLized Web pages: some information is maintained by individuals themselves (such as their research interests), other information is maintained by the division director (such as project assignments), and some information is maintained at the institute level (such as office assignments); this relieved the division's administrative assistant from manually maintaining everyone's interests [4]. WebScripter has also been used externally, for example to maintain a Semantic Web tools list [6], and a DAML publications list [7]. WebScripter can be downloaded from [8].

As of the time of writing, one issue we encountered is that there is not really that much interesting, continuously updated RDF Schema, much less DAML or OWL, available on the Web today.[2] What made the original Web take off was that there was an immediate incentive for producers to use the technology because it was an easy way to publish information. We currently see little motivation for Web page authors to put work into producing RDF in addition to

[2] One notable exception are RSS 1.0 [9] headline exchange files such as *slashdot.org/slashdot.rdf*.

their regular HTML pages (as others have noted also, maybe most eloquently in [17]). In this section, we will describe a novel WebScripter application, which not only makes use of WebScripter but also incentivizes web authors to produce RDF contents.

3.1 Using WebScripter for Collaborative Semantic Weblogs

We're now in the process of applying WebScripter technology to weblogs. Such a "blog" is a "frequent, chronological publication of personal thoughts and Web links" [10]. It is an easy way to publish a piece of information with a single click. It is estimated that there were already from 500,000 to 1 million web bloggers since 1999 and that the number of bloggers is still expanding rapidly [12].

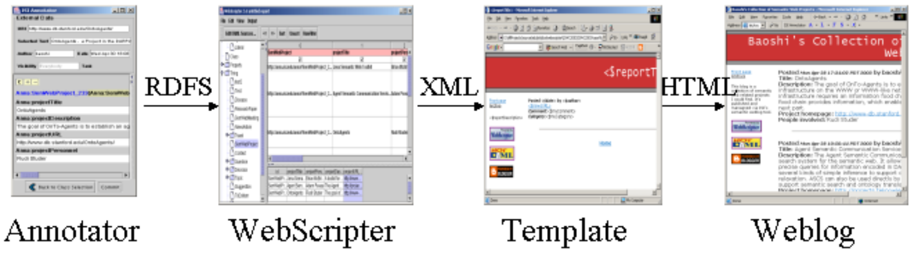
Weblogs, though very popular nowadays, have no semantic structure, which brings about several shortcomings. A weblog is a collection of posts. Each post is a segment of natural language text in HTML format. There is no metadata describing individual posts. The lack of semantic structure makes it difficult to organize weblogs. For example, suppose you have two weblogs, one is about the Semantic Web, the other about Java programming. Hence, you would want to add a discussion of Jena [22] to both weblogs. However, currently you have to either only add it to one weblog, or copy the same content over to the other blog, neither of which is satisfactory. The lack of semantics also rules out queries like "what are posts about Java programming for the Semantic Web". Weblogs quickly become clumsy for information retrieval as the volume of data increases because their only native indexing is by reverse chronological order.

WebScripter, coupled with an easy-to-use metadata publishing tool, could greatly enhance the functionality of weblogs. A weblog with no data other than the text and the entry date of a post can be viewed as a two-column WebScripter report. Additional columns can then be used for additional semantic mark-up about the posts. WebScripter by itself supports RDFS report authoring not original RDFS data entry. Thus, we also developed ISI Annotator, which provides an easy-to-use way to produce metadata. Annotator lets users define their own classes and properties for describing their posts.[3] The posts users publish are in RDFS format and can thus be post-processed by WebScripter (or any other RDFS tool, of course).

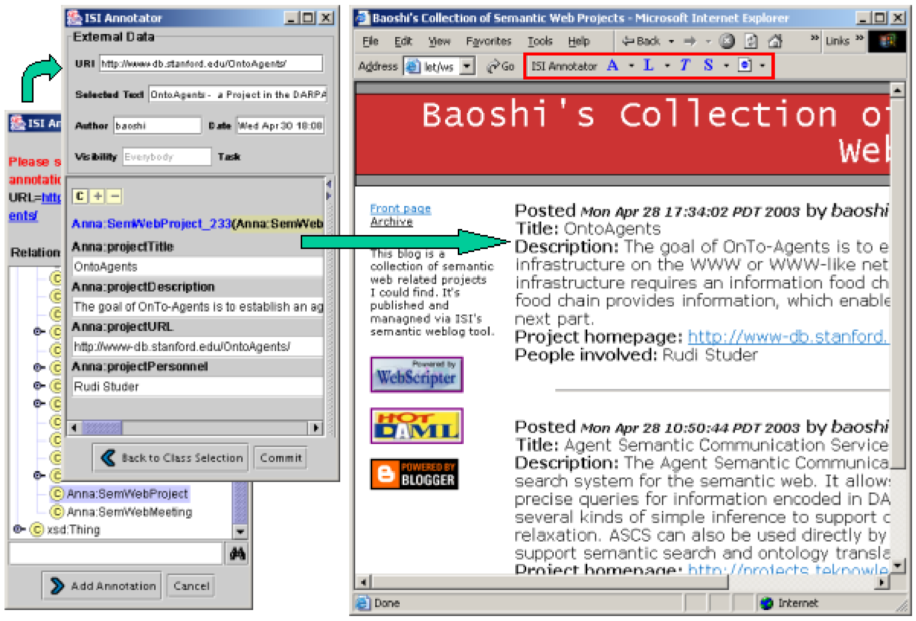
Figure 8 shows the chain of producing RDFS content in Annotator, post-processing it as a WebScripter report, and finally presenting it as a Web page with (invisible) RDFS mark-up via a stylesheet. Note that it is perfectly possible to define multiple WebScripter reports of that data, say one that only picks up posts marked as Java-related or as personal.

One common phenomenon in weblogs is cross-referencing between web bloggers with similar interests, which form a community where they read each other's blog, comment and share blogs. When a user builds a new report by aligning someone else's semantic weblog with her own, the immediate result is that she

[3] Annotator also allows users to annotate existing Web pages with RDFS, hence its name, but that is not further discussed in this paper.



(a)



URI http://www-db.stanford.edu/OntoAgents/
Selected Text OntoAgents - a Project in the DARPA
Author baoshi Date Wed Apr 30 18:08
Visibility Everybody

Anna:SemWebProject_233(Anna:SemWeb
Anna:projectTitle
OntoAgents
Anna:projectDescription
The goal of OnTo-Agents is to establish an ag
Anna:projectURL
http://www-db.stanford.edu/OntoAgents/
Anna:projectPersonnel
Rudi Studer

Please s
annotatic
URL=http
ents/

Anna:SemWebProject
Anna:SemWebMeeting
xsd:Thing

Baoshi's Collection o
Wel

Front page
Archive

This blog is a collection of semantic web related projects I could find. It's published and managned via ISI's semantic weblog tool.

Posted *Mon Apr 28 17:34:02 PDT 2003* by *baoshi*
Title: OntoAgents
Description: The goal of OnTo-Agents is to e
infrastructure on the WWW or WWW-like net
infrastructure requires an information food ch
food chain provides information, which enable
next part.
Project homepage: http://www-db.stanford.
People involved: Rudi Studer

Posted *Mon Apr 28 10:50:44 PDT 2003* by *baoshi*
Title: Agent Semantic Communication Service
Description: The Agent Semantic Communica
search system for the semantic web. It allow
precise queries for information encoded in DA
several kinds of simple inference to support c
relaxation. ASCS can also be used directly by
support semantic search and ontology transla
Project homepage: http://projects.teknowle

(b)

Fig. 8. Semantic Weblogs: (a) ISI's Annotator Tool can produce RDFS that is then run through a WebScripter report and produces the final Web page that contains both human and machine-readable mark-up. (b) ISI's Annotator(on the left) and the web page of a semantic weblog

now will have other's semantic weblog content in her semantic weblog, but there are other important implications. First, the implicit alignment axioms inferred from her alignment would benefit other bloggers in doing the same work. Second, by adding aother's semantic weblog to hers, the user implies that there are shared interests. Such information could in turn facilitate the discovery of bloggers with similar interests, thus expediting the forming of a blogger community.

4 Related Work

WebScripter's approach to ontology alignment is extreme: terms from different ontologies are always assumed to mean different things by default, and all ontology mapping is done by humans (implicitly, by putting them into the same column of a report) – in that sense, there is no automated inference.

This is similar in spirit to Gio Wiederhold's mediation approach to ontology interoperation [27], which also assumes that terms from different ontologies never mean the same thing unless committees of integration experts say they are. WebScripter pushes that concept to the brink by replacing the experts with ordinary users that may not even be aware of their implicit ontology alignment contributions. (Note, however, that we cannot yet proof that this collective alignment data is indeed a useful source for automatic ontology alignment on an Internet scale – we lack sufficient data from distributed WebScripter use to make that claim.)

Most schema matching techniques (see [26] for a survey) take a semi-automated approach to ontology interoperation: the system guesses likely matches between terms of two separately conceived ontologies, a human expert knowledgeable about the semantics of both ontologies then verifies the inferences, possibly using a graphical user interface. Such guesses can be based on name and structure similarity in schemas (ONION [24] , PROMPT [25]), plus data instances (LSD [14] GLUE [15]), or integrated use of different techniques (CUPID [21], COMA [13]). In WebScripter human users rely purely on the *data instances* to decide what collates and what does not (because they are just not expert enough to analyze the abstractions). That being said, incorporating the above techniques into WebScripter would clearly be beneficial if the rate of correct guesses is sufficiently high. Unlike other schema matching techniques, WebScripter is not suitable for alignment tasks where the only information available are the two schemas to be matched (without any instance data for the schemas). Rather, it relies on shared reuse and reasoning with other users' implicit and often imprecise ontology alignments.

OBSERVER [23], SIMS [11], TSIMMIS [16] and the Information Manifold [19] are all systems for querying multiple data sources of different schemata in a uniform way; however, they all rely on human experts to devise the ontological mappings between the sources to our knowledge. This is because they mediate between structured dynamic data sources (such as SQL/ODBC sources) without run-time human involvement where a higher level of precision is required to make the interoperation work. In contrast, WebScripter is targeted towards mediating between different ontologies in RDF-based Web pages with run-time human involvement, where the need for precision in the translation is naturally lower.

Another difference between WebScripter and the above systems is its philosophy of creating the Semantic Web while using it. Our vision and hope is that semantics can emerge from large-scale collaborative use of numerous, previously isolated ontologies.

5 Discussion

More appropriate Semantics for Alignment Axioms Strictly speaking, the alignment axioms inferred from WebScripter should not be DAML or OWL equivalence statements, which are very strong claims: "x equivalent to y" means everything that applies to x also applies to y and vice versa. The WebScripter alignment axioms should really just imply equivalence within some context – two concepts are equivalent for the purposes of the report that some individual created. Thus on the one hand, we need more work on judging the likelihood of two contexts being compatible, possibly according to report contents, user profile, or user report-authoring history. On the other hand, we need to evaluate how generic an alignment axiom is. We would propose a more grass-roots, somewhat Darwinian theory. The theory is that relatively small groups will use these axioms to align their semantics and eventually you will get alignment among groups of groups, and for a few things this will lead to alignment among really big groups. Nevertheless, we believe approximate alignment axioms in the spirit of Hovy's "generally accoiated with" links [18] will be a common phenomenon in the Semantic Web as it is unlikely that concepts casually developed by end users would meet as strong requirements as implied by the DAML or OWL equivalency statements.

Will WebScripter scale? The current implementation of WebScripter does not scale well because (a) all alignment axioms are stored in and retrieved from a single central server, and because (b) all processing occurs in the main memory of the user's machine. (Essentially, we chose not to worry about scalability until we got to a compelling application and a substantial number of users.) We believe that scalability for the former could be achieved with a server-farm approach as demonstrated by e.g. Google and Yahoo, or with a peer-2-peer approach to distribute equivalency data across the users' machines. The latter is already being addressed by RDF toolkits that can connect to back-end databases, such as Jena [22].

End-User Control over Auto-Alignment Our current end-user interface for alignment (see Section 2.2) is unlikely to scale well for large numbers of alignment axioms. We see the following solutions to this problem (which are not mutually exclusive). *Social Filtering.* One approach would be to keep track of the authors of alignment axioms as well as the users of alignment axioms; this would enable users to say "I want to use the same equivalency data that Jim and Chris are using", a nicely implicit way to limit equivalencies to e.g. the accounting context if they are co-workers in accounting, without having to more formally define the context, which is a more abstract and difficult task. This would also allow cautious users to express "I am willing to use equivalency data that at least ten others are using" (which addresses the erroneous-alignment problem but not the context-mismatch problem). *Finer-Grained Control in the User Interface.* It would be nice to have a display (see Table 2) of the available equivalency information that presents more than just the equivalent URIs as we currently do (lower window of Figure 7). A related question is how users can lock out ill-intentioned sources of alignment axioms, addressing the general question of trust management on the Semantic Web. In a future stage of WebScripter, a Google-like social filtering mechanism and a Spam-blocking-like collaborative blacklist could help.

Class	Hops	Origin	Author	Rows	Date	Users
Person	1	stanford.e...	Smith	235	10/6/02	12
Employee	1	stanford.e...	Smith	57	10/6/02	6
Staff	1	stanford.e...	Smith	697	10/6/02	0
Member	2	www.isi.e...	Chen	15	3/4/01	17
Person	2	cmu.edu/...	Miller	973	12/7/01	4
Member	2	cmu.edu/...	Miller	107	12/7/01	9

Table 2. Sketch of a graphical user interface for better end-user control over alignment.

6 Conclusion

As an easy-to-use report authoring tool, WebScripter has proven its usefulness in several applications. As far as we know, WebScripter is currently the only interactive report generator for RDFS content. The most exciting application of WebScripter, as a collaborative ontology translation tool, has yet to prove its effectiveness due to the small number of casual Semantic Web users. Nevertheless, we are excited about this new approach to ontology alignment sharing. The key difference we see between "traditional" ontology translation and our approach is that non-experts perform all of the translation - but potentially on a global scale, leveraging each other's work.

7 Acknowledgments

We gratefully acknowledge DARPA DAML program funding for WebScripter under contract number F30602-00-2-0576. We thank Bob MacGregor and Stefan Decker for helpful discussions.

References

1. http://reliant.teknowledge.com/DAML.
2. http://www.daml.org/crawler.
3. http://annotation.semanticweb.org/iswc/ documents.html.
4. http://www.isi.edu/divisions/div2/. Click on People.
5. http://www.dfki.uni-kl.de/ruleml.
6. http://tools.semanticweb.org.
7. http://www.daml.org/publications/cite.html.
8. http://www.isi.edu/webscripter.

9. http://web.resource.org/rss/1.0.
10. http://www.marketingterms.com/dictionary/blog/.
11. Y. Arens, C. Knoblock, and W.-M. Shen. Query reformulation for dynamic information integration. *Intelligent Information Systems*, 6(2-3):99–130, 1996.
12. W. W. Conhaim. Blogging – what is it?, May 2002. http://www.infotoday.com/LU/may02/conhaim.htm.
13. H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, 2002.
14. A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, 2001.
15. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map ontologies on the semantic web. In *The Eleventh International World Wide Web Conference*, 2002.
16. H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: data models and languages. *Intelligent Information Systems*, 8(2):117–32, 1997.
17. S. Haustein and J. Pleumann. Easing participation in the semantic web. In *WWW-2002 Semantic Web Workshop*, Honolulu, Hawaii, May 7 2002.
18. E. Hovy. Using an ontology to simplify data access. *Communications of the ACM*, 46(1):47–49, 2003.
19. A. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Intelligent Information Systems*, 5(2):121–43, 1995.
20. J. Lopez and P. Szekely. Web page adaptation for universal access. In *UAHCI-2001 Conference on Universal Access in Human Computer Interaction*, pages 690–694, New Orleans, 2001. Lawrence Erlbaum Associates, Mahwah, NJ.
21. J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.
22. B. McBride. Jena: Implementing the rdf model and syntax specification. Technical report, Hewlett-Packard, 2000. http://www-uk.hpl.hp.com/people/bwm/papers/20001221-paper/.
23. E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. OBSERVER: an approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–71, 2000.
24. P. Mitra and G. Wiederhold. An algebra for semantic interoperability of information sources. In *2nd Annual IEEE International Symposium on Bioinformatics and Bioengineering*, pages 174–82, Bethesda, MD, USA, November 4-6 2001.
25. N. F. Noy and M. A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *17th National Conference on AI*, 2000.
26. E. Rahm and P. Bernstein. On matching schemas automatically. Technical report, Microsoft Research, Redmon, WA, 2001. MSR-TR-2001-17.
27. G. Wiederhold. Interoperation, mediation, and ontologies. In *International Symposium on Fifth Generation Computer Systems, Workshop on Heterogeneous Cooperative Knowledge-Bases*, volume W3, pages 33–48. ICOT, Tokyo, Japan, December 1994.