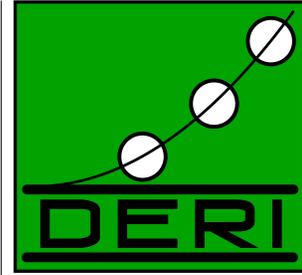


DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE



SEMANTIC WEB PUBLISHING WITH DRUPAL

Stéphane Corlosquet Richard Cyganiak
Axel Polleres Stefan Decker

DERI TECHNICAL REPORT 2009-04-30
APRIL 2009

DERI Ireland
University Road
Galway
IRELAND
www.deri.ie

DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE

DERI TECHNICAL REPORT

DERI TECHNICAL REPORT 2009-04-30, APRIL 2009

SEMANTIC WEB PUBLISHING WITH DRUPAL

Stéphane Corlosquet, Richard Cyganiak, Axel Polleres, Stefan Decker¹,

Abstract. Getting Semantic Web data and annotations into and out of end-user applications is one of the many challenges of making the Semantic Web fly. While linked data principles are slowly taking off and being adopted by a small number of sites and gradually more exporters to link existing Web content to the Web of data, we still find ourselves in the cold start phase. While producing Web content has become easy for end users by content management systems (CMS), blogging tools, etc. the problem of enabling end users to produce semantic Web content persists. In this short paper, we propose a framework for a one-click solution to lift the huge amounts of Web content residing in CMS systems to the Web of Data. We tackle one of the most popular CMS systems nowadays, Drupal, where we enable site administrators to export their site content model and data to the Web of Data without requiring extensive knowledge on Semantic Web technologies. We have developed a Drupal module that maps the inherent site structure of a typical Drupal site to a lightweight ontology that we call the Site Vocabulary, and that exposes site data directly in RDFa. As such, this simple solution would not link to existing Semantic Web data, since site vocabularies exist decoupled from the widely used vocabularies in the Linked data cloud. To close this gap, we have incorporated an easy-to-use, fail-safe ontology import and reuse mechanism in our module, that allows site administrators – with a few clicks – to link their site vocabulary, and thus their site data, to existing, widely used vocabularies on the Web. In whole, our approach shall help to bootstrap the Semantic Web by leveraging the huge amounts of data in CMS. In approaching CMS site administrators rather than end users, we tackle the problem of adding semantics where we consider it easiest: Semantics are fixed at design time based on the site structure, whereafter end users entering data produce Linked data automatically. We have evaluated our approach in user experiments and report on deployment of our module in the Drupal community.

Keywords: Content Management, Linked Data, RDFa, Drupal.

¹Digital Enterprise Research Institute Galway, National University of Ireland Galway
University Road Galway, Ireland E-mail: firstname.lastname@deri.org

Acknowledgements: This work was supported by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT, SWWS, Esperonto, and h-TechSight; by Science Foundation Ireland under the DERI-Lion project; and by the Vienna city government under the CoOperate program.

Copyright © 2009 by the authors

Contents

1	Motivation	1
2	Related works	2
3	Drupal: An Popular CMS	2
4	From Content Models to Site Vocabularies	4
5	Adhering to linked data principles	5
6	Mapping to Existing Ontologies	5
7	Evaluation	8
7.1	Acceptance	8
7.2	Publishing effort	8
7.3	Motivation and Benefits	9
8	Conclusions and Outlook	9

1 Motivation

Since the late 90ies and early 2000s a paradigm shift has taken place in Web publishing towards a separation of data (content) and structure (mainly layout). The first ideas to have the data represented in a way which allows its reuse in various ways and not only HTML, emerged in parallel in various systems such as Typo3 (1997), Plone (1999), webML [13]. These open source systems and their commercial counterparts are nowadays typically subsumed under the common term *content management systems* (CMS).

While it is worthwhile to mention that the first of these systems appeared at around the same time as Semantic Web technologies emerged, with RDF [14] being standardized in 1999, the development of CMS and Semantic Web technologies have gone largely separate parts. Semantic Web technologies have matured to the point where they are increasingly being deployed on the Web. Large amounts of RDF data can now be accessed over the Web as *Linked Data*. This data is used by a variety of clients, such as RDF data mashups that integrate information from various sources, search engines that allow structured queries across multiple sites and datasets, and data browsers that present a site's content in new ways. But the traditional Web still dwarfs this emerging Web of Data and in fact, the HTML Web – boosted by technologies such as CMS – is still growing at much faster pace as the Semantic Web, see also [3].

When thinking about how to re-link the HTML Web and the Semantic Web of Linked data. Thus, the task of “RDFizing” existing web sites, particularly the provision of means to produce Linked data form sites that are anyways already highly structured, is of high potential benefit. The recently finished RDFa [1] standard additionally supports this by allowing RDF to be embedded directly into existing HTML pages.

Alongside with the success story of CMS, we observe that a large portion of Web data is de facto already available in such highly structured form, though typically still exposed in plain HTML to the outside: Content Management Systems are typically built on a structured model of the domain of the site which is reflected in both the underlying database, but – also and often more accurately – in the content types defined in the CMS system itself by a site administrator. A typical example of a state-of-the art CMS is Drupal, which has been gaining popularity recently by its rapidly growing user community, openness and modular design.

In this paper we present a module for Drupal which allows to link back this content model to the Web of Data. After installing this module on an existing Drupal site, the content types and fields typically defined by a Drupal site administrator using Drupal's Content Construction Kit (CCK) will be automatically exposed in terms of classes and properties of a canonical OWL ontology, the *site vocabulary*. Likewise, the site data itself becomes available as RDFa embedded in the live Web site following linked data principles without additional effort for the administrator.

While this off-the-shelf solution already potentially makes any Drupal site which installs our module amenable to Semantic Web tools such as RDF crawlers, search engines, and SPARQL engines to search and query the site, this is only a first step. For real linkage to the Web of data, we need to enable mappings of the site vocabulary to existing, widely used ontologies. To this end, our module extends CCK by additional means to import and map to existing vocabularies. To keep the burden low for site administrators who normally will have little interest in learning the details of RDF and description logics, we support save-reuse, i.e. only allow to link content types and fields to existing classes and properties in a fail-safe manner, that is we avoid the modification of existing ontologies as well as the introduction of potential inconsistencies in our user interface as far as possible.

2 Related works

Let us explain how our approach differs from previous attempts to link CMSs with the Semantic Web and why we believe that it is more adequate.

Staab et al. [18] proposed – at a stage where CMSs in their current form were still in their infancy – what could be conceived as an ontology based Content management system. The idea was to separate Web Content from domain models, very similar to the core ideas that made the success of CMSs. Similarly, Ontowebber [12], promoted the creation of Websites out of ontologies with a system built on a native RDF repository. These approaches though focus on ontologies and ontology design themselves as a backbone to organise data on Web sites. On the contrary, content management systems typically support very lightweight structuring of information items supported by User interfaces limited to the needed functionality to structure data for Web presentation, rather than full ontology editing tools. Moreover, current CMS typically rely on off-the-shelf relational databases to store data on the backend, rather than RDF repositories. Our module for Drupal thus has a very orthogonal goal: Rather than building an ontology-based CMS we aim to extract and link ontologies from the content models and within the tools typical site administrators are familiar with nowadays. We believe that this strategy – taking users from where they are in an unobtrusive way – to be the key enabler to leverage Semantic Web technologies with low entry barriers.

Somewhat closer to our approach are the ideas presented by Stojanovic et al. [19], where a relational database schema underlying a CMS is mapped to RDF Schema or Frame Logic. A more recent approach – Triplify [3] – follows a similar path, providing a generic mapping tool for lifting relational databases into Linked Data by dedicated mappings from a relational Schema to RDF. It should be mentioned that Triplify even provides some predefined mappings to wrap Drupal sites' backend databases into RDF. Again, our approach is significantly different. Due to the flexible nature of Drupal the underlying database used to store the data underlying a CMS based site does not necessarily reflect the sites content model and its constraints. Thus, the relational schema might vary between different versions of Drupal on the one hand, and on the other hand, the effects of changing the content model to the database schema underneath by the site administrator are not always obvious. Actually, a good part of the success of content management systems is grounded precisely in the fact that site administrators do not need to delve into details of the underlying database system or schema. Our approach works on a more abstract level (Drupal's CCK) directly in the API the site administrator is used to, where all the information about the content model is available, which we believe to model the information structure more adequately than the underlying database schema. The site administrator does not need to know anything about the underlying database to create mappings to existing RDF vocabularies or expose RDFa on her site.

3 Drupal: An Popular CMS

Before we start explaining our work, let us briefly introduce the main concepts behind Drupal. Drupal¹ is a popular open-source content management system. It is among the top three open-source CMS products in terms of market share [17]. Drupal facilitates the creation of web sites by handling many aspects of site maintenance, such as data workflow, access control, user accounts, and the encoding and storage of data in the database.

As typical for CMS, a *site administrator* initially sets up a site by installing the core Drupal web application and choosing from a large collection of *modules* that add specific functionality to the site, such as

¹<http://drupal.org/>

improved user interface, enhanced search, various export formats, extended statistics and so on. Site administrators need a fair bit of technical knowledge to choose and configure modules, but usually do not write code; this is done by module developers instead. After the site has been set up, Drupal allows non-technical users to add content and handle routine maintenance of the site.

Each item of content in Drupal is called a *node*. Nodes usually correspond more or less directly to the pages of a site. Nodes can be created, edited and deleted by content authors. Some modules extend the nodes, for example a taxonomy module allows assignment of nodes to categories, and a comment module adds blog-style comment boxes to each node.

The *Content Construction Kit (CCK)* is one of the most popular and powerful modules used on Drupal sites. It allows the site administrator to define types of nodes, called *content types*, and to define *fields* for each content type. Fields can be of different kinds such as plain text fields, dates, email addresses, file uploads, or references to other nodes. Additional kinds of fields can be added via modules. When defining content types and fields, the site administrator has to provide the following information: *ID*,²*label*, and *description* for content types and fields. Additionally, CCK allows to specify the following constraints on fields:

- *Cardinality*: fields can be optional or required, and may have a maximum cardinality.
- *Domain*: fields can be shared among one or more content types.
- *Range*: fields can be of type text, integer, decimal, float, date, file attachment, or node reference; fields of type node reference can be restricted to nodes of specific content types; fields of type text can be restricted to a fixed list of text values.

As a running example, we choose a cheese review web site,³ the site administrator might define content types such as *Cheese*, *Review*, and *Country of Origin*. The *Cheese* type might have fields such as *description*, *picture*, or *source of milk*. Figure 1 shows the Administrator interface to edit a content type “Cheese” in Drupal along with some constraints defined on the “source of milk” field.



Figure 1: Administrating a content type in Drupal’s CCK (left) and defining constraints on a field (right).

Thus, site administrators use CCK to define a site-specific content model, which is then used by content authors to populate the site. The focus of our work is to expose (i) the CCK site content model as an OWL ontology that reflects the site structure which the designer had in mind and (ii) the site content as RDF data using this ontology.

²a string of lower case alphanumeric characters.

³Demo-site available at <http://drupal.deri.ie/cheese/>

4 From Content Models to Site Vocabularies

We have implemented a Drupal module that enhances Drupal's CCK with the ability to auto-generate RDF classes and properties for all content types and fields. We build a so-called *site vocabulary*, i.e., an RDFS/OWL ontology which describes the content types and fields used in the data model as classes and properties. The field and type names are extracted from field and type *IDs* from CCK, such that – following common conventions – fields are assigned a property name starting with a lower case character, and content types are assigned a class name starting with an upper case character. Field and content type labels and descriptions are likewise exported as `rdfs:labels` and `rdfs:comments`. Here goes a typical content type and field definition extracted from CCK into RDFS:

```
site:MyType a rdfs:Class; rdfs:label "My Type";
  rdfs:comment "My Type description";
site:myField1 a rdf:Property; rdfs:label "my field";
  rdfs:comment "My field description";
```

Likewise, field constraints on from CCK are reflected in the site vocabulary:

Cardinalities are mapped to cardinality restrictions in OWL, i.e. *required* fields are restricted to `owl:minCardinality 1`. whereas fields with a maximum cardinality n are restricted to `owl:maxCardinality n`. For instance, if we assume that each *Cheese* is required to have *name* field, and at most 5 *reviews*, these constraints in CCK would be exported to OWL as follows.

```
site:Cheese a rdfs:Class; rdfs:subClassOf
  [ a owl:Restriction; owl:onProperty site:name;
    owl:minCardinality 1],
  [ a owl:Restriction; owl:onProperty site:review;
    owl:maxCardinality 5].
```

Domains are reflected by `rdfs:domain` constraints. Here, fields used by a single type can be modeled by a simple `rdfs:domain` triple. For instance, assuming that the *review* field for *Cheeses* is not shared with any other content type in the current content model, we can simply write:

```
site:review rdfs:domain site:Cheese.
```

CCK fields shared among several types have the union of all types sharing the field as their domain. E.g., since *Cheese*, *Country* and *Region* share the *name* field, the site vocabulary contains

```
site:englishName rdfs:domain
  [owl:unionOf (site:Cheese site:Country site:Region)].
```

Ranges of fields are analogously encoded by `rdfs:range` triples. Additionally, we distinguish between fields of range text, integer, decimal, float, or date, and those referring to file attachments, or node references. We declare the former as `owl:DatatypeProperty` and assign the datatypes supported in Drupal with their respective XSD datatypes, i.e. *text* → `xs:string`, *integer* → `xs:integer`, *decimal* → `xs:decimal`, *float* → `xs:float`, or *date* → `xs:date`. For instance, the text field *name* is reflected in the site vocabulary as:

```
site:name rdfs:range xs:string; a owl:DatatypeProperty .
```



```
@prefix site: <http://drupal.deri.ie/cheese/ns#>

<http://drupal.deri.ie/cheese/node/3>
  rdf:type site:Cheese;
  site:origin "Germany";
  site:name "Handkäse";
  site:source_milk "cow";
  site:description "Handkäse (literally: hand cheeses ...)";
  site:review <http://drupal.deri.ie/cheese/node/8> .
```

Figure 2: Drupal page with embedded RDF auto-generated from CCK content model.

Fields that range over texts restricted to a fixed list of text values will be assigned a respective enumerated class of values using `owl:Datatypes`, e.g. *source of milk* is modeled as

```
site:sourceOfMilk a owl:DatatypeProperty; rdfs:range
[ a owl:Datatype; owl:oneOf ( "cow", "goat", "sheep" ) ].
```

Finally, fields that range over file attachments (which get a URI in Drupal) or node reference, are declared of type `owl:ObjectProperty`. Fields ranging over more than one content type are reflected in the site vocabulary by `owl:unionOf`. E.g., *origin* may be a Country or a Region, respectively.

```
site:origin a owl:ObjectProperty; rdfs:range
[ owl:unionOf ( site:Country site:Region ) ].
```

5 Adhering to linked data principles

Following the conventions mentioned in the previous section, the site vocabulary is generated and published automatically at the site URL under the default namespace `http://siteurl/ns#`, which we denoted by the namespace prefix `site:` in the examples before. Likewise, any Drupal page on a site will by our module be annotated with RDFa triples that dereference terms of this site vocabulary as classes and properties linking Drupal content nodes as subjects and objects. We are inline with the Linked data principles and best practices [5, 4] as we provide resolvable HTTP URIs for all resources: Each of the pages also represents a node of a certain content type in the CCK content model. That is, in our model, each node becomes an RDF resource, and the HTML Web page describing the node is enriched with RDFa [1] that reflect the links in the content model. By this design, any Drupal site usign our module is off-the-shelf amenable to any existing tool that can consume Linked Data style RDF content.

Figure 2 shows a page in our cheese example site with embedded RDF dereferencing the site ontology along with the embedded auto-generated RDF data.

6 Mapping to Existing Ontologies

While the functionality we have described so far fits Drupal sites well into the Linked data world, so far, we have created nothing more than an isolated ontology from th existing site content model. However, this benefits of this exercise remain limited, unless we additionally allow linking the site vocabulary to existing vocabularies and ontologies populating the Semantic Web.

For instance, instead of just exposing the *Cheese* type as a class in the site vocabulary, you might want to reuse a class in an existing ontology, such as for instance `ov:Cheese` from the OpenVocab⁴

⁴<http://open.vocab.org/terms/>

vocabulary which some other publishers on the web already used. Or likewise, we may wish to state that a *Review* is actually a `foaf:Document` from the widely used FOAF⁵ ontology, or that a cheese linked to its Description by the widely used `dc:description` property, from Dublin Core⁶, etc.

To this end, our module adds a new tab “Manage RDF mappings” to the content type administration panel of CCK for managing such mappings to existing ontologies, cf. Figure 3. After importing ontologies via a simple dialog, the terms of these ontologies can be mapped to the Cheese type and its fields.

The mapped terms will result in `rdfs:subClassOf` and `rdfs:subPropertyOf` mappings being added to the site vocabulary as well as extra triples using the mapped terms being exposed in the RDF embedded in the page.

Step 1: Import ontology

Import vocabulary

Vocabulary URI:

Prefix:

Step 2: Define mappings

Cheese Edit Manage fields Manage RDF mappings Display fields

Specify the RDF class of this content type. You can also map the CCK fields to existing RDF properties.

RDF class:

Label	Name	Type	RDF property
Name	Node module form.		foaf:label
Picture	field_picture	Image	foaf:description
Description	field_description	Text	dc:description
Source of milk	field_source_milk	Text	dc:description
Country of origin	field_country_origin	Text	dcp:license

Mappings added to the site vocabulary:

```
site:Cheese rdfs:subClassOf ov:Cheese .
site:name rdfs:subPropertyOf rdfs:label .
site:description rdfs:subPropertyOf dc:description .
```

Triples added to the exposed RDF of Fig. 2 (italic):

```
<http://drupal.deri.ie/cheese/node/3>
rdf:type site:Cheese; rdfs:type ov:Cheese;
site:origin "Germany";
site:name "Handkäse"; rdfs:label "Handkäse";
site:source_milk "cow";
site:description "Handkäse (literally: hand cheeses ...)";
dc:description "Handkäse (literally: hand cheeses ...)";
dcp:license "The URL of an RDF description of the license the software is";
site:review <http://drupal.deri.ie/cheese/node/8> .
```

Figure 3: RDF mappings management through the Drupal interface.

As mentioned before, the interface checks certain features to avoid inconsistent site data/vocabularies, such as checking consistency of cardinalities of reused properties with the field cardinalities.

To map the site data model to existing ontologies, the site administrator first imports the ontology or vocabulary. We assume that it has been created using a tool such as Protégé⁷, OpenVocab⁸, or Neologism⁹, and published somewhere on the Web in RDF Schema or OWL format.

For every content type and field, the site administrator can choose a class or property it should be mapped to, see Figure 3. Such mappings are expressed as `rdfs:subclass` and `rdfs:subproperty` relationships. For instance, if the field *englishName* that was used on types *Cheese* and *Country* is mapped to FOAF’s `foaf:name`, then a triple

```
site:englishName rdfs:subPropertyOf foaf:Name .
```

will be added to the site ontology. Additionally, we allow inverse reuse of existing properties. E.g., assume the site administrator imports a vocabulary `ex:` that defines a relation between Countries/Regions and goods that this region/country produces via the property `ex:produces`. Our user interface also allows to relate fields to the inverse of imported properties. For instance, the *origin* field could be related to `ex:produces` in such an inverse manner, resulting in

```
site:origin rdfs:subPropertyOf
  [ owl:inverseOf ex:produces ] .
```

⁵<http://xmlns.com/foaf/0.1/>

⁶<http://purl.org/dc/elements/1.1/>

⁷<http://protege.stanford.edu/>

⁸<http://open.vocab.org/>

⁹<http://neologism.deri.ie/>

being added to the site vocabulary.

Likewise, for assigning an existing class to a content type we use subclassing, e.g.

```
site:Review rdfs:subClassOf foaf:Document .
```

Subclassing and the use of subproperties is a simple way of minimizing unintended conflicts between the semantics of local vocabulary and public terms. Per OWL semantics, constraints imposed on the local term by the content model/site vocabulary will thus not apply to the public term. This ensures *safe vocabulary re-use*, i.e. avoids what is sometimes referred to as “Ontology Hijacking” [2].

Intuitively, safe reuse means that a vocabulary importing another one does not modify the meaning of the imported vocabulary or “hijack” instance data of the imported vocabulary.

Let us assume that we would, on the contrary, directly use the imported properties and classes in the site vocabulary. That would cause problems. For instance, the export of the content model as described in the previous section contains the triple `site:englishName a owl:DatatypeProperty`. Would we have used `foaf:name` directly here, we would have changed the FOAF vocabulary, which, by itself doesn’t explicitly declare `foaf:name a owl:DatatypeLiteral`. Direct reuse of existing classes in the site vocabulary would raise similar issues.

We emphasize, however, that reuse of `rdfs:subClass`, `rdfs:subProperty` as well as inverse `rdfs:subProperty` relations alone is not sufficient to guarantee consistency of the site vocabulary. While the following proposition intuitively holds, this is no longer true as soon as external vocabularies are imported.

Proposition 1 *A site vocabulary that does not import any external ontologies is always consistent.*

Nevertheless, in case of importing external properties and classes, consistency can no longer be guaranteed without further restrictions, even if both the site vocabularies and the imported ontology were consistent. This is easily illustrated by some examples.

Example 1 Assume that one would inversely assign the `foaf:homepage` property to `site:review`. Given that now `foaf:homepage` is an inverseFunctional property, this would imply functionality of `site:review`, i.e. that each cheese would have at most one review. This would, possibly result in strange side effects on the site instance data such as yielding reviews for cheeses with two or more reviews equal. ◇

In other cases, contradicting cardinalities could even make site instance data inconsistent. Likewise the inverse reuse of Datatype Properties is problematic. To address this problem, when displaying external properties assignable to fields in the content model (see Figure 3) our tool extending CCK could make several restrictions such as not displaying properties with cardinality restrictions attached that do not comply with those specified in CCK for the respective field.

We emphasize that we do not aim to deploy a full OWL reasoner to detect all inconsistencies possibly introduced by ontology reuse in our system. The identification of syntactic fragments of OWL, which are safely usable in our tool without the need to deploy a fully-fledged OWL (DL) reasoner (i.e., which features used in the imported ontologies require which level of expressiveness for detecting possible inconsistencies by reuse) is on our agenda.

Our ultimate goal is a tool which allows the site administrator to import classes and properties from existing ontologies in a fail-safe way such that no possible inconsistencies may be possibly introduced by the extended CCK editor and later filling of the site with content.

The reason why we are reluctant of deploying full reasoning services is that we want our tool to fit in the Drupal ecosystem as a normal module that is installable on a site without the need to install separate external software components such as a DL reasoner. Wherefore we restrict ourselves to lightweight reasoning by applying heuristics such as the detection of cardinality violations mentioned above. An alternative path would be the invocation of a reasoning service deployed as a web-accessible service.

It must be stressed that the whole mapping step is optional, and the main benefit of the Web of Data – exposing site data for re-use by third parties – is realized by the default mapping.

7 Evaluation

We exposed our hypothesis and general rationale that ease-of-use and a one-click solution to export Linked data from CMSs will boost the Semantic Web to a small user evaluation. What we aimed to proof is that linking a site to existing vocabularies by use of our Drupal module does not impose a significant burden to site administrators and the benefits of exposing Semantic Web data such as searchability may outweigh this negligible extra effort. To this end, In this section we evaluate the usage of the implementation we created and the extra effort required by our approach.

7.1 Acceptance

In order to make this implementation available to as many developers as possible, we have released the RDF CCK module on drupal.org. Since its release November 2008, the RDF CCK module has reached a number of 45 installations¹⁰ as shown in the Figure 4. As a concrete example, the module is currently being tested and will be deployed in the next version of the Science Collaboration Framework (SCF) platform by Harvard Med School [9].

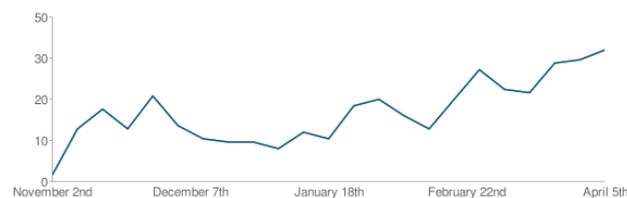


Figure 4: Evolution of the number of installations of RDF CCK since its release.

7.2 Publishing effort

A formal evaluation was carried out on 10 students who were more or less familiar with the Semantic Web, but not with the Drupal User Interface. They were asked to set up a predefined content model using a guide. Each major step of the process was timed in order to be exploited in a statistical analysis.

The content model used in this evaluation was composed of 2 content types *Article* and *Editor* which included 5 and 4 fields, respectively.

¹⁰according to the Usage statistics for RDF CCK page at <http://drupal.org/project/usage/rdfcck>

Initial setup of the content model The first major step of the evaluation was to setup the content model the way any site administrator would on a typical Drupal site. The CCK User Interface was used here. The time each student took to execute this step is materialized in orange in the Figure 5.

RDF mappings setup The second step we evaluated is the RDF mappings. The pool of 10 students was divided into two 5 people groups: the group A (user 1 to 5) had to decide which mappings the most appropriate and the group B (users 6 to 10) was given a list of predefined mappings.

According to our measures and as shown on Figure 5, the extra step required to have a fully mapped content model represent less than half of the time of the initial setup.

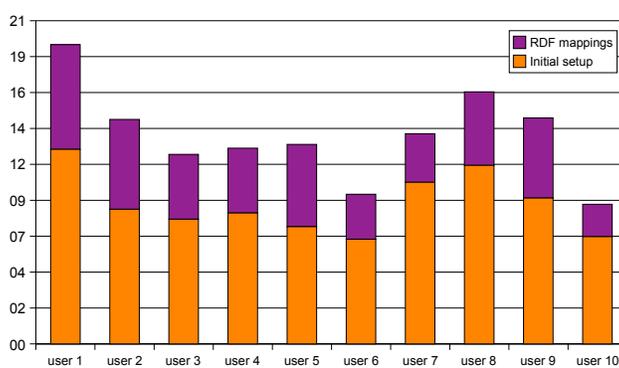


Figure 5: Comparison between initial setup and RDF mappings.

On average, the extra time spent on specifying the mappings took 58% of the initial setup for the group A and 39% for the group B.

7.3 Motivation and Benefits

In the recent years, website administrators have been encouraged to publish sitemaps in order to promote their site on the web. Similarly, semantic sitemaps[8] allow sites to describe semantic web datasets and how to best consume it. Our approach requires an extra effort which has similar benefits. Moreover, most of this extra effort is eased by the User Interface and can be done automatically.

Yahoo! SearchMonkey, Exhibit, Visinav, Sindice[16].

Summarizing, our findings show that whereas linking to external vocabularies was experienced easy by our test group, the main effort and time consumed was actually in deciding to which properties and classes to link, that is which ontologies should be reused how. Inspired by this finding we put on our agenda more investigation on how we can support non-Semantic-Web-savvy users in finding the “right” classes and properties for their needs. To this end, we aim to extend search functionalities of Semantic Web search engines such as Sindice [16] or SWSE [11] by search and ranking functionalities to find and link widely used classes and ontologies for linking them to your content model, based on keywords or field names.

8 Conclusions and Outlook

In this paper we have presented an extension to Drupal’s CCK that allows to link existing and newly deployed Drupal sites to the Web of data by a few clicks. It auto-exports the site content model of a Drupal

site to an ontology that is published following common best practices for ontology publication and enables the exposure of Drupal site content as RDFa out-of-the-box.

The site vocabulary uses a fragment of the available OWL constructs which is sufficient to reflect the site content model's structure in a machine-readable manner and is guaranteed to be consistent. In order to link to existing vocabularies from the Semantic Web, we allow ontology import and linkage of existing properties and classes by subclass/subproperty relations.

The system we have implemented is a working prototype¹¹. The module used for the prototype and discussed in this paper is available on drupal.org¹². The "infection" of promising power-user communities such as the rapidly growing Drupal site administrator and developer groups is in our opinion a key in boosting Semantic Web technologies.

Next steps include the extension of RDF/OWL export for Drupal to other modules such as for instance the Taxonomy module for tag hierarchies usable in Drupal, which we plan to expose as RDF using SKOS[15].

Our solution shall provide easy-to-use, unobtrusive RDF exposure in a way general enough for a variety of Drupal sites, thus potentially contributing significantly to the further population of the Web of data with high-quality RDF data.

Acknowledgments This work was supported by SFI grant No. SFI/08/CE/I1380 (Lion-2)

References

- [1] B. Adida, M. Birbeck, S. McCarron, S. Pemberton (eds.), RDFa in XHTML: Syntax and processing, W3C Rec., <http://www.w3.org/TR/rdfa-syntax/> (Oct. 2008).
- [2] A. Hogan, A. Harth, A. Polleres, SAOR: Authoritative reasoning for the web, ASWC 2008.
- [3] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, D. Aumüller, Triplify - lightweight linked data publication from relational databases, WWW 2009.
- [4] D. Berrueta, J. P. (eds.), Best practice recipes for publishing rdf vocabularies, W3C Working Group Note, <http://www.w3.org/TR/swbp-vocab-pub/> (Aug. 2008).
- [5] C. Bizer, T. Heath, K. Idehen, T. Berners-Lee (eds.), Linked Data on the Web (LDOW) 2008.
- [6] D. Brickley, R. Guha (eds.), RDF vocabulary description language 1.0: RDF Schema, W3C Rec., <http://www.w3.org/TR/rdf-schema/> (Feb. 2004).
- [7] D. Brickley, L. Miller, FOAF Vocabulary Specification, <http://xmlns.com/foaf/0.1/> (Nov. 2007).
- [8] R. Cyganiak, H. Stenzhorn, R. Delbru, S. Decker, G. Tummarello, Semantic sitemaps: Efficient and flexible access to datasets on the semantic web, ESWC 2008.
- [9] S. Das, L. Girard, T. Green, L. Weitzman, A. Lewis-Bowen, T. Clark, Building biomedical web communities using a semantically aware content management system, Briefings in Bioinformatics (2009) 10(2):129–38.

¹¹A demo site is online at <http://drupal.deri.ie/cheese/>.

¹²<http://drupal.org/project/rdfcck>

- [10] M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein, OWL Web Ontology Language Reference, W3C Rec. (Feb. 2004).
- [11] A. Harth, J. Umbrich, S. Decker, Multicrawler: A pipelined architecture for crawling and indexing semantic web data, ISWC 2006.
- [12] Y. Jin, S. Decker, G. Wiederhold, Ontowebber: Model-driven ontology-based web site management, SWWS 2001.
- [13] Y. Jin, S. Xu, S. Decker, G. Wiederhold, Managing web sites with ontowebber, in: C. S. Jensen, K. G. Jeffery, J. Pokorný, S. Saltenis, E. Bertino, K. Böhm, M. Jarke (eds.), EDBT 2002.
- [14] O. Lassila, R. Swick (eds.), Resource Description Framework (RDF) Model and Syntax Specification, W3C Rec., <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/> (Feb. 1999).
- [15] A. Miles, S. Bechhofer (eds.), SKOS Simple knowledge organization system reference, W3C Cand. Rec., <http://www.w3.org/TR/2009/CR-skos-reference-20090317/> (Mar. 2009).
- [16] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, G. Tummarello, Sindice.com: A document-oriented lookup index for open linked data, Int'l Journal of Metadata, Semantics & Ontologies 3 (1).
- [17] R. Shreves, Open source cms market share, White paper, Water & Stone, <http://waterandstone.com/downloads/2008OpenSourceCMSMarketSurvey.pdf>.
- [18] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H. P. Schnurr, R. Studer, Y. Sure, Semantic community web portals, Computer Networks 33 (1-6) (2000) 473 – 491.
- [19] L. Stojanovic, N. Stojanovic, R. Volz, Migrating data-intensive web sites into the semantic web, ACM SAC 2002x.