Journal of Library Metadata, 16:228–249, 2016
Published with license by Taylor & Francis
ISSN: 1938-6389 print / 1937-5034 online
DOI: 10.1080/19386389.2016.1258897

Routledge
Taylor & Francis Group

Combining the Best of Both Worlds: A Semantic Web Book Mashup as a Linked Data Service Over CMS Infrastructure

AIKATERINI K. KALOU, DIMITRIOS A. KOUTSOMITROPOULOS, and GEORGIA D. SOLOMOU
High Performance Information Systems Laboratory (HPCLab), Computer Engineering and Informatics Department, University of Patras, Patras-Rio, Greece

We propose a proof-of-concept of how the best of the Web world (RESTful services, Semantic technologies, and CMSs) can be successfully incorporated in the form of a semantic mashup. This mashup enriches data from various Web APIs with semantics to produce personalized book recommendations and to make them available as Linked Data. It is shown that this approach not only leaves reasoning expressiveness and effective ontology management uncompromised but comes to their benefit. It also bears potential applications for libraries and cataloguing services that can take advantage of mashup merging, semantic personalization, and ranking of book records and reviews.

KEYWORDS search mashup, Linked Data, Web APIs, RESTful services, CMS, personalization.

A *mashup* (Koschmider, Torres, & Pelechano, 2009) is a Web-based application that is created by combining and processing on-line third party resources, for example, APIs and Linked Data. With the prevalence of the Semantic Web, traditional mashups are evolved into semantic mashups that consume data from interlinked data sources on the cloud.

Address correspondence to Dimitrios A. Koutsomitropoulos, University of Patras, School of Engineering, Computer Engineering and Informatics Dpt., High Performance Information Systems Laboratory, Building B, 26500 Patra-Rio, Greece. E-mail: kotsomit@hpclab.ceid.upatras.gr

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/wjlm.

The Linked Open Data (LOD) project[1] has successfully brought a great amount of data to the Web. The availability of interlinked data sets encourages developers to reuse content on the Web and alleviates the need to discover various data sources. In the case of semantic mashups, contribution to the LOD effort can come by appropriately combining data from Web APIs with semantics and then providing them as Linked Data.

As is often the case with any Semantic Web application, semantic mashup development usually puts too much effort into the bottom-up construction of elaborate, knowledge intensive set-ups. This kind of application often dwells on high-end reasoning services, efficient rule processing, and scalability over voluminous data, thus hardly leaving any room for traditional Web development.

This gap can be bridged by traditional Web content management systems (CMSs), which offer an up-to-date and tailored Web infrastructure and leave more room for the designer to concentrate on successful content production and delivery rather than technical details. As CMSs form the spearhead of Web 2.0, it might then feel natural to employ them as a basis for Semantic Web applications, but this presents a series of challenges that are not always straightforward to overcome.

In this paper, we therefore propose how such applications and CMSs can be integrated by presenting Books@HPClab, a semantic mashup application, which we purposely establish on top of the Drupal CMS. Books@HPClab (Kalou, Pomonis, Koutsomitropoulos, & Papatheodorou, 2010; Solomou, Kalou, Koutsomitropoulos, & Papatheodorou, 2011) was initially developed from scratch and offers personalization features to users searching for books from various data sources. The key concept of this mashup is that it gathers information from Amazon and Half.com Web APIs, enriches them with semantics according to an ontology (in this case, the BookShop ontology[2]) and then employs OWL 2 reasoning to infer matching preferences. Book metadata, being now represented as triples in OWL/RDF, can also be linked to other resources, thus becoming more reusable and, effectively, more sharable on the LOD cloud. We also wrap additional RESTful web service functionality around our semantic mashup as a means for other applications to consume and exchange Linked Data without manual intervention.

As a result, Books@HPClab can offer personal recommendations to users based on a set of certain criteria. Compared to conventional online bookshops, the semantic book mashup is capable of integrating and combining data from multiple sources and of applying intelligent matching techniques on top of them. Finally, the application is surrounded by a number of automated Web services that can help its reuse within larger settings, including digital libraries, open educational repositories, online course providers, and learning systems.

The following text is organized as follows: In "Background Technologies", we survey the background technologies semantic mashups are based on and their relationship with bibliographic standards. We also discuss the desirable properties of CMSs that make them suitable as a basis for developing Semantic Web applications. In "Ontology Design", we describe the BookShop ontology and its alignment with the latest bibliographic frameworks. In "System Design", we explain how we proceeded with the actual integration and discuss how we addressed the problems arising in this process by putting particular focus on the data workflow and reasoner integration. In "Book Mashup", we specify the provision of Linked Data and the development of corresponding services. In "Use Cases", we illustrate the features and the functionality of our deployment, now completely reengineered over Drupal; further, we outline its potential applications in the library world. "Evaluation" includes the methodology and results of the evaluation process, in terms of effectiveness and efficiency. Finally, Section 8 summarizes our conclusions and suggests future work.

BACKGROUND TECHNOLOGIES

In this section, we give an overview of the technology stack that is exploited for the implementation of the work presented in this paper.

A *conventional mashup* is a Web-based application that is created by combining one or more on-line third party resources and finally making the existing, Web-based data more useful (Huajun et al., 2009). The core element behind building such applications is the *REST Web APIs* (*Representational State Transfer Application Programming Interface*). In plain words, they can be thought of as a set of rules that describe the way to fetch data from the database behind a website. Many well-known websites such as Facebook, Twitter, LinkedIn, and Google offer their data via this mechanism, preserving the privacy and security terms.

Even though a mashup is tightly related to Web APIs for their development, there are a few limitations such as (a) structural diversity of data sources, (b) nonuniform semantics of aggregated data, and (c) lack of extensibility and interoperability. Toward overcoming these limitations, the next generation of mashups, called *semantic mashups*, are making their appearance. A semantic mashup (Kalou & Koutsomitropoulos, 2015) constitutes a Web mashup that employs Semantic Web technologies and ideas (ontology, OWL, RDF, Linked Data) in any part of its design, architecture, functionality, or presentation levels.

The Semantic Web, proposed by Tim Berners-Lee (inspirator of the Web) and propagated by the World Wide Web Consortium (W3C) is not a separate Web but an extension of the current one, in which information is

given well-defined meaning, better enabling computers and people to work in cooperation (Berners-Lee et al. 2006).

Ontology is the "backbone" of the Semantic Web and formally defines a common set of terms used to describe and represent the basic concepts in a domain and the relationships among them (Berners-Lee, Hendler, & Lassila, 2001). At the core of the Semantic Web architecture stack, there appears *reasoning* (Antoniou et al., 2005), the key component for the derivation of facts unexpressed explicitly in an ontology. OWL and RDF (Manola & Miller, 2004) are among the most widely used formal semantic languages.

Linked Data (Bizer, Heath, & Berners-Lee, 2009) focuses on the creation of typed links between different data sources by using the Web. Linked Data, from a technical aspect, are data with explicitly defined meaning, published on the Web in a machine-readable format (data in RDF), which are linked to other external data sets and can also be linked to form external data sets. The result of this connection is the Web of Linked Data.

The notion of the Semantic Web and, especially, Linked Data seems to have a fruitful application in the field of bibliographic standards and library cataloguing. More and more, there are initiatives and standards that have grown out of research results and that are gradually turning into industrial practice. Such standards that utilize the ontology structure in order to efficiently express their resource metadata schemata include Dublin Core (IEEE LTSC, 2008), FRBR (Manguinhas, Freire, Machado, & Borbinha, 2012; Takhirov, Aalberg, Duchateau, & Žumer, 2012), and others. As a step beyond conceptualization, the principles of Linked Data are widely adopted so as to increase the discovery and visibility of available bibliographic resources. Take for example BIBFRAME,[3] the replacement for MARC, which is entirely focused on Linked Data as its basic data model.

Finally, a *CMS* (*Content Management System*) (Patel, Rathod, & Prajapati, 2011) is an application (more likely Web based) that provides capabilities for multiple users with different permission levels to manage all or a section of content, data, or information of a website project, or internet/intranet application. This sense of managing content refers to creating, editing, archiving, publishing, collaborating on, reporting, or distributing website content, data, and information.

A typical CMS generally comes with the ability to help and facilitate the user, even the nontechnical one, in various ways. It always ensures a set of core features, already implemented and resolved, including a front-end interface, a user management facility, dynamic content management, caching support and so on (Patel, Rathod, & Prajapati, 2011).

Features such as these, which contemporary CMSs unsparingly offer, are exactly the ones sometimes neglected by Semantic Web applications. Recognizing the benefits of incorporating semantic technologies within traditional CMSs, many initiatives have been taken to accomplish this objective (Ngonga Ngomo, Heino, Lyko, Speck, & Kaltenböck, 2011). For instance,

the well-known Drupal 7 CMS provides users with significant functionalities of expressing data in RDF and RDFa and querying over SPARQL endpoints (Clark, 2011; Corlosquet & Cyganiak, 2009).

In the case of our work, we chose to integrate Books@HPClab within the core of Drupal 7 (Tomlinson, 2010). Regardless of Drupal's semantic character, other significant advantages such as flexibility and scalability make it stand out from the large pool of CMSs. Additionally Drupal has been used before as a basis for offering Linked Data services. In our approach, we maintain all the processes that concern the transformation of data to semantically enhanced data and vice versa, as well as the storage of these data, within the core of the Drupal infrastructure. The reasoning procedure is an autonomous service, accessible via RESTful interfaces.

ONTOLOGY DESIGN

Given the very nature of our application, a state of the art review of cataloguing bibliographic resources was fundamental and influential for the ontology design process. In particular, FRBR and BIBFRAME, built upon the entity-relationship model and both expressed in RDF, seem to be applicable in the case of our semantic mashup, covering its requirements and scope.

Bibliographic Metadata Over the Semantic Web

Over the years, the library community realized that the insufficiencies of the relational database and document format technologies dominating the existing bibliographic data representation standards could be limited by the incorporation of Linked Data principles. To this end, the emerging library standards are being redefined by embracing a multi-entity model with varying levels of abstraction. FRBR, RDA, and BIBFRAME are among the most representative new approaches that have been published in RDF.

FRBR (Functional Requirements for Bibliographic Records) (Davis, D'Arcus, & Newman, 2005) is a conceptual entity-relationship model that comprises bibliographic entities (grouped in three categories), their attributes, and the relationships between them. Group 1 entities refer to intellectual or artistic products (*Work, Expression, Manifestation, Item*), from the more abstract level to the more concrete level. Group 2 entities refer to people (*Person*) and/or corporate bodies (*Corporate Body*) that create, publish, or preserve the Group 1 entities. Group 3 entities are used as topics in the Group 1 entities (*Concept, Object, Event*, and *Place*).

RDA (Resource Description and Access)[4] has been developed as a set of new cataloguing guidelines, designed exclusively for the digital world and more precisely for the Web world. These new guidelines contain a set of practical instructions based on FRBR so as to describe all types of resources,

including online ones. Descriptions created using RDA will be usable in Web-based catalogues and other resource discovery services.

The most recent and pure-RDF bibliographic framework, BIBFRAME 2.0,[5] intends to accommodate many bibliographic formats and data models and the only prerequisite stated is the use of Linked Data technologies. *Work, Instance*, and *Item* are the three main classes of this model. The class *Work* reflects the "conceptual substance of the cataloguing item" and seems to be semantically closer to the FRBR *Work* and *Expression* entities. The class *Instance* reflects "an individual, material embodiment of the Work" and seems to be akin to the FRBR *Manifestation* entity. The class *Item* is an actual copy of an *Instance* reflecting information such as its location, shelf mark, and barcode. Next, there are three additional key concepts (*Agents, Subjects* and *Events*) that have relationships to the core classes. The class *Agents* captures people, organizations, and other groups, associated with a *Work* or *Instance* through roles such as author, editor, or illustrator. *Subjects* represent the topics, places, temporal expressions, events, and so forth of the *Work*. Finally, the class *Events* reflects occurrences, the recording of which may be the content of a *Work*.

BookShop Ontology

Considering the kind of metadata offered by Amazon and Half.com responses and the main classes of BIBFRAME and FRBR, focused on the entity of *Book*, we designed the core ontology BookShop shown partially in Figure 1. BookShop contains five main classes *Book, Author, Offer, User*, and *Modality*.

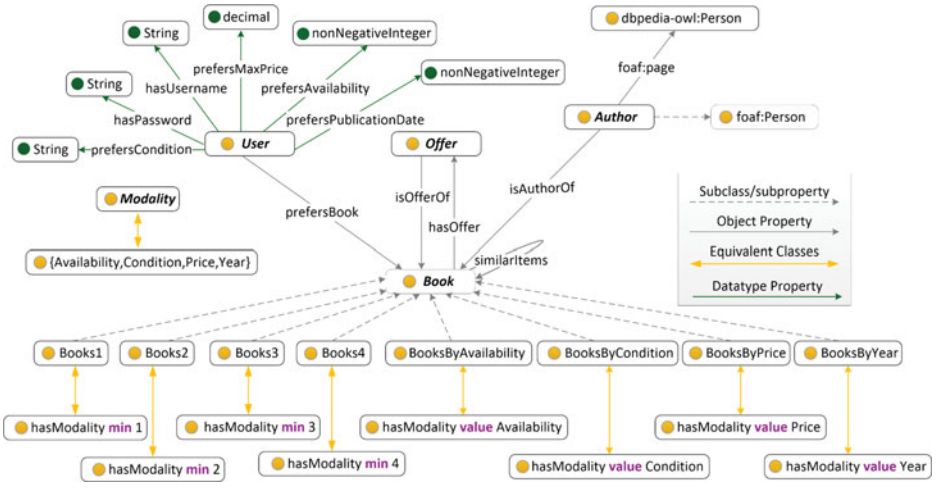

FIGURE 1 BookShop ontology.

In our ontology, the class *User* is meant to express user profiles. We capture the preferences of each user in this class, such as preferable condition, preferable minimum availability, preferable minimum publication year, and preferable maximum price (preference criteria). All this data about users are represented as data type properties.

The class *Book* represents all book items that are gathered from Amazon and Half.com sales markets. This class is enriched with relations such as title, publisher, dimensions, ISBN, publication year, number of pages, format, rating, images in various sizes, and a URL—corresponding to the Amazon online bookstore—so as to describe the instances of this class. All these relations are captured in the ontology as data type properties.

For our purposes, keeping a record of each author's name and surname suffices. Therefore, we decided to define the class *Author* as a subclass of the class *Person*, which is included in the FOAF (Friend of a friend) ontology. FOAF provides a unified way to describe persons, expressing their interests, their activities, and their relations to other people and objects.

Items for sale on Amazon and Half.com can be sold by more than one seller for different prices and in different conditions ("new" or "used"). Thus, any item—any book in this case—is associated with an offer. An offer is a combination of price, condition, and vendor/seller. The concept of an offer is represented by the class *Offer*.

A reasoner is responsible for entailing which books match what criteria in the current user profile and classifies them accordingly (*BooksByAvailability, BooksBycondition, BooksByPrice, BooksByYear*). The type of a matched criterion is represented by the members of the *Modality* class. Given the cardinality restrictions on the *hasModality* property, the books are finally classified depending on the number of satisfied preference criteria (*Books…Books4*). For example, the *Books1* class contains all the books that match at least one of the preference criteria.

Even though the BookShop ontology is more abstract and not as detailed as the aforementioned bibliographic models, it is clear that direct mappings with most of the entities are possible. In brief, the class *Book* of our BookShop ontology for example is a conflation of the main core classes of BIBFRAME 2.0, *Work, Instance*, and *Item* as well as the Group 1 entities of FRBR. Next, the *Person* class of the FRBR model and the *Agents* class of BIBFRAME 2.0 could be related to the *Author* class, specified by the BookShop ontology.

SYSTEM DESIGN AND INTEGRATION

We present the overall design of our application and its interaction with all necessary external and embedded components. We also describe thoroughly

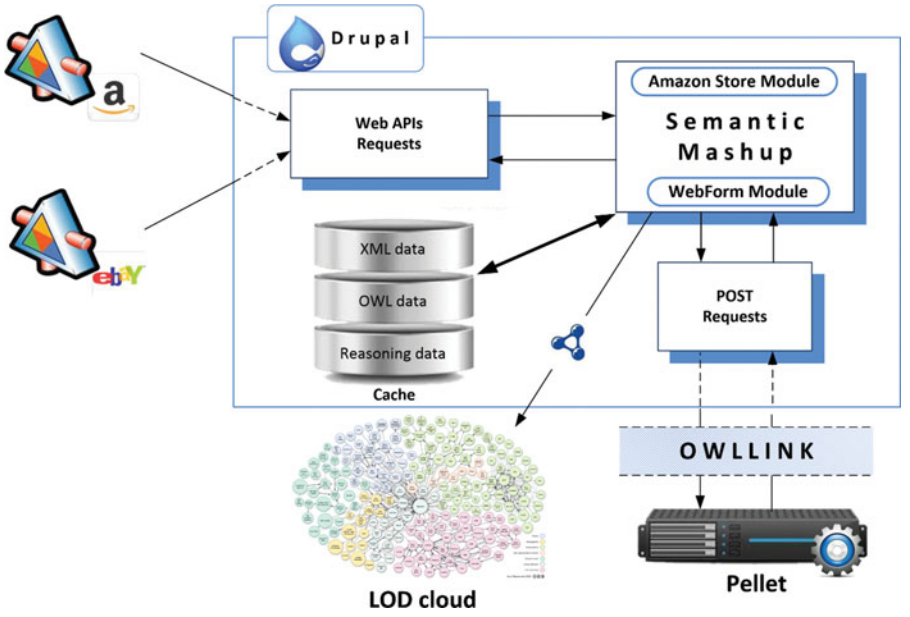

FIGURE 2 Architecture and communication flow for integrating Semantic Mashup with Drupal.

the main issues we had to deal with and how we addressed each one of them.

Architecture and Integration Challenges

The modular philosophy of a CMS allows us to extend its capabilities with ready-made modules and to reuse them for our purposes. To this end, we utilize the *AmazonStore module* that offers an attractive wrapper and front-end for the Amazon Web API. We have extended this module so as to include support for Half.com as well. We also make use of the *WebForm module*, which supports form-based data collection and which we use as the initiating point for constructing user profiles. The WebForm architecture of our re-engineered mashup is illustrated in Figure 2.

In order to re-engineer our semantic mashup on top of Drupal so as to leverage all the aforementioned features, we encountered a series of challenges, originating from the fact that CMSs are usually not semantics aware. Although latest versions of Drupal offer some inherent semantic features (Bratsas, Bamidis, Dimou, Antoniou, & Ioannidis, 2012), in our implementation we needed to put a strong focus on reasoning and ontology management as well as data interlinking, which is beyond Drupal's current capability (or any other CMS's, for that matter). All these issues are analyzed in the following subsections.

Data Collection and Storage

In the context of our application, with the term *data* we mean the consolidation of user profiles, information collected from the external sources and ontological data before and after the reasoning process. In this subsection, we review in detail the data collection and storage workflow and all the existing Drupal modules that we have exploited to this end.

Regarding user profiles construction, user preferences are collected using Web forms, designed with the aid of the WebForm module. A unique ID is assigned to each user. In addition to user preferences, each user has to set a unique password and username, as well as his or her e-mail address so notifications from the application can be sent and received. All this user information is stored in tables of the relational database.

In order to perform reasoning, however, these preferences have to be translated into semantically rich expressions, which form the ontological profile of each user. In our case, we retrieve user preferences from the database and then we construct the profile on the fly by mapping preferences to a set of OWL 2 expressions.

Once our application completes the search process at Amazon, it starts searching Half.com; for each book returned by Amazon, we find additional offers that may be available at Half.com. We use the *Half.com shopping Web Services* and particularly, the *FindHalfProducts* operation. The interaction with the Half.com shopping API is based also on the REST protocol and the exchange of URL requests and XML files responses. By augmenting the data storage policy of AmazonStore module, we save the Amazon XML results, enriched with additional book offers from Half.com, in the *XML data cache* (see Figure 2).

Next, search results need to be transformed into the OWL word in order to enable inferences. This conversion adheres to our BookShop ontology schema and is achieved via XSLT. The transformed ontological data are cached in the *OWL data cache*. In order to achieve personalization, OWL data as well as the ontological user profile are sent to the remote reasoning service. Finally, the inferred knowledge is stored at the *reasoning cache*.

An algorithm (shown in Table 1) is responsible for synchronizing between the caches, which, apart from checking for repeating queries, additionally expunges the reasoning cache whenever a user updates his or her profile. Note that the cache can be flushed after a configurable amount of time (in this case, 24 hours). A profile update initiated by a user causes the removal from the cache of all reasoning results related to the particular profile u—that is, $\mathcal{R} \rightarrow \mathcal{R} / \{r_{*,u}\}$, where * denotes all o_q.

The adoption of the database caching and data replication strategy allows CMS modules to remain oblivious to the ontology data and lets them operate on their own data cache. This caching idea, which is also carried

TABLE 1 Algorithm for the Synchronization of Data Storage

$\mathcal{B}$: XML book data cache, b_q: XML book data for query q

$\mathcal{O}$: Ontological book data cache, o_q: ontological book data for query q

$\mathcal{R}$: Reasoner results cache, $r_{o_q,u}$: reasoner results for o_q and user profile u

if $\{b_q\} \notin \mathcal{B}$
then $b_q \rightarrow$ **get_amazon_data** (q)
$b_q \rightarrow b_q \cup$ **get_ebay_data** (q)
$\mathcal{B} \rightarrow \mathcal{B} \cup \{b_q\}$
$o_q \rightarrow$ **triplify** (b_q)
$\mathcal{O} \rightarrow \mathcal{O} \cup \{o_q\}$
$r_{o_q,u} \rightarrow$ **invoke_reasoner** (o_q, u)
$\mathcal{R} \rightarrow \mathcal{R} \cup \{r_{o_q,u}\}$
return $r_{o_q,u}$

if $\{b_q\} \subseteq \mathcal{B}$, $\{o_q\} \subseteq \mathcal{O}$ **and** $r_{o_q,u} \notin \mathcal{R}$
//since b_q is in $\mathcal{B}$, o_q will always be in $\mathcal{O}$
then $r_{o_q,u} \rightarrow$ **invoke_reasoner** (o_q, u)
$\mathcal{R} \rightarrow \mathcal{R} \cup \{r_{o_q,u}\}$
return $r_{o_q,u}$
if $\{b_q\} \subseteq \mathcal{B}$, $\{o_q\} \subseteq \mathcal{O}$ **and** $r_{o_q,u} \subseteq \mathcal{R}$
then return $r_{o_q,u}$

over to reasoning results, actually improves the effective reasoning throughput by keeping reasoner engagement to a minimum.

Reasoning Integration

In our implementation, we use OWLlink (Liebig, Luther, Noppens, & Wessel, 2011) as the reasoner communication protocol of choice and its implementation, the OWLlink API (Noppens, Luther, & Liebig, 2010) that helps us deploy a true, three-tier architecture. OWLlink offers a consistent way of transmitting data to and receiving responses from the most popular Semantic Web reasoners over HTTP. Potential communication overhead that may be introduced with this approach can be alleviated by freeing up resources as a consequence of delegating computationally hard reasoning tasks to another tier (Koutsomitropoulos, Solomou, Pomonis, Aggelopoulos, & Papatheodorou, 2010). Moreover, Drupal offers us generic function implementations that can be used to wrap and construct HTTP requests, like `drupal_http_request`. Messages are encoded in XML format and Pellet is used as the inference engine of choice.

The interaction between the OWLlink server and our client application consists of four main request-response messages. First, we allocate a knowledge base (KB) within the OWLlink server by sending a CreateKB request. The unique user ID is assigned as an identifier to the KB in order to logically separate knowledge bases under the same reasoner. In the same message, we embed a LoadOntologies request so as to load the BookShop ontology schema into the given KB by reading the ontology file.

Next, we add the ontological user profile and the OWL data results for a specific query by sending two distinct Tell requests to the OWLlink server. At this point user preferences are fetched from the database and are

TABLE 2 Interaction With OWLlink Server

	No. 1	No. 2	No. 3	No. 4
Request	CreateKB kb = [*User_ID*] LoadOntologies IRI = [*BookShop ontology*]	Tell *preferences* BooksByPrice ≡ ∃hasOffer. (∃offerPrice.[≤ *user_pref*]) BooksBy-Condition … BooksBy-Availability … BooksBy-Year …	Tell data OWL Book data from cache (query results)	GetFlattenedInstances direct = ″true″ class IRI = *{Books1, Books2, Books3, Books4}* ReleaseKB kb = [*User_ID*]
Response	Response-Message OK	Response-Message OK	Response-Message OK	SetofIndividuals {*1..4*} NamedIndividuals IRI = [*Book resource URL*]

used to construct the ontological user profile on the fly, which amounts to a set of OWL 2 restrictions (see Table 2).Both user profile and OWL data are encoded in OWL/XML syntax. In order to get the inferred knowledge from the reasoner, we send a GetFlattenedInstances request. Its purpose is to retrieve all books that satisfy up to four preference criteria (instances of *Books1, Books2, Books3*, and *Books4* classes). The `direct = true` parameter ensures that the above sets will be mutually disjointed—that is, they will include only unique book instances. Finally, the KB is destroyed by issuing a ReleaseKB request within the same message. Table 2 summarizes all the messages that are exchanged between our application and the OWLlink server.

BOOK MASHUP AS A SERVICE

We describe meticulously how we provide and transform all the mashed up data into Linked Data, making them available to human users and machines. We also explain the way that these Linked Data can be integrated with the LOD cloud. In addition, we describe the upgrade of Books@HPClab application with REST operations.

Linked Data Service

To publish Linked Data for book items, we mint HTTP URIs using the pattern shown in Figure 3, which is based on the application's namespace. First, each book item is uniquely identified by a single URI, describing the item itself. Then, we assign to each book another URI that describes the item

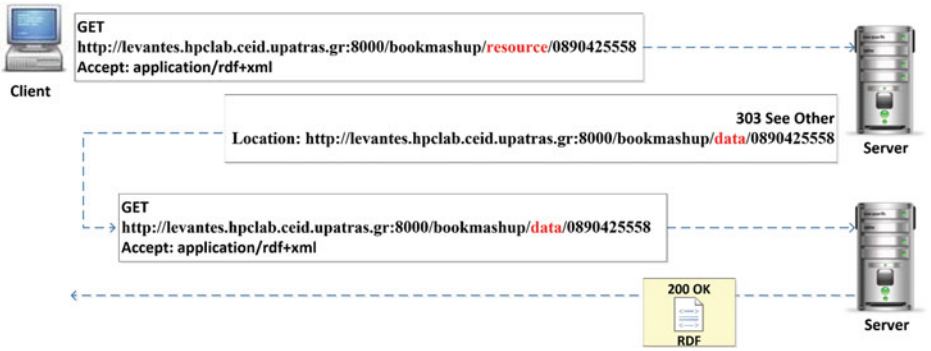

FIGURE 3 A complete example of content negotiation.

and has an HTML representation, appropriate for consumption by humans. Next, another URI is given to the book item to describe it and provides an RDF/XML representation for machine readability.

To associate our data with other data sets on the Web, we interlink our entities with others by adding RDF external links. More precisely, in the case of book offers, relationship links are added so as to point to the bookstore origin. We also inject DBpedia HTTP URIs into author RDF descriptions originally available from the Web APIs. Figure 4 depicts an excerpt of published RDF data with the external RDF links.

FIGURE 4 Interlinking the data set of Books@HPClab with external data sets.

TABLE 3 HTTP Request Fomat in the Context of Books@HPClab Service

POST Request	Response
URI: *bookmashup/bookmashup /service* **HTTP version:** HTTP v1.1 Request Header: *Accept Header (application/rdf+xml)* *Content Type (application/xml)* **Request Body:** *<?xml version = "1.0"?>* *<user_profile>* *<publication_year>2015 </publication_year>* *<max_price>15.50</max_ price><availability>48 </availability>* *<condition>New</condition>* *<keyword>Big Data</keyword>* *</user_profile>*	*<rdf:RDF base =* "http://www.hpclab.ceid.upatras.gr/BookShop#"> <bs:User rdf:about = "12299"> <bs:prefersBook> <bs:Book rdf:about = "http://levantes.hpclab.ceid.upatras.gr:8000/bookmashup/resource/111866146X"> <bs:hasRating rdf: datatype = "http://www.w3.org/2001/XMLSchema nonNegativeInteger"⟩2 </bs:hasRating></bs:Book> </bs:prefersBook> … </bs:prefersBook> </bs:User> </rdf:RDF>

Book Mashup as a RESTful Service

This approach of publishing and sharing linked data of individual book records can be extended toward performing queries and fetching search results also using REST. The HTTP protocol can naturally cater to dynamic requests and updates of linked data sets.

Therefore, the search functionality, which was previously available only via the user interface, can now be leveraged by other applications, hereafter without the need for human intervention. To achieve this, we need to communicate the search keyword as well as the search constraints (the *user profile*) using POST (left column of Table 3).

A sample response is shown in the right side of Table 3. The response includes resolvable URIs to book records matching the requested criteria. Note that each of these URIs will be negotiated and resolved accordingly, as per the resolution scheme of Figure 3. In addition, for every record, the number of matching criteria (the book *rating*) is also included as an integer, thus making the response self-contained. Each POST request is assigned a random, unique ID and this is retained in the response message.

USE CASES

In this section, we demonstrate the whole functionality of the Books@HPClab application. We also point out the integration possibilities of the application with others and especially how it could work with learning services.

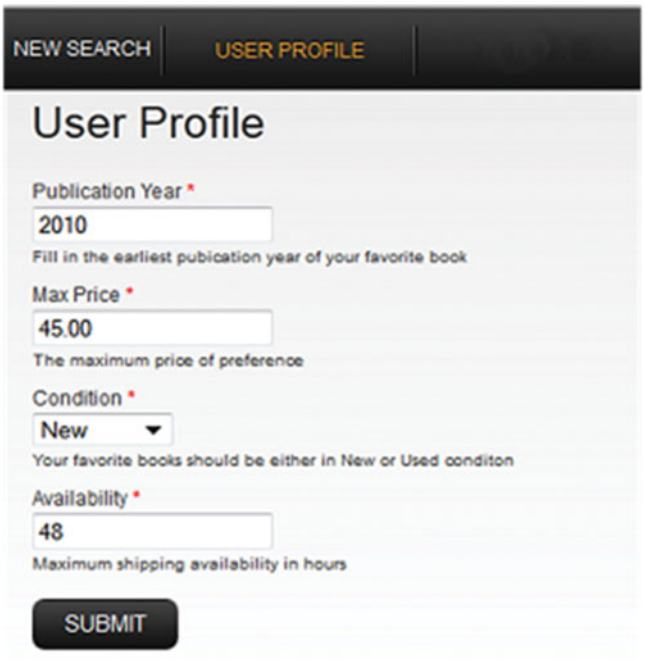

FIGURE 5 Collecting user preferences.

Usage Scenario

After successful authorization, logged users can set their profile using the WebForm module. The form fields correspond to user preferences and include book condition ("new" or "used"), maximum book price, earliest publication year, and maximum availability (Figure 5). A user can update his or her profile at any time. Note also that if a user does not define preferences, the application behaves as a standard book mashup and the reasoner is never engaged.

The "available offers" pull-down list also includes Half.com offers that have been mashed up and linked with Amazon book data. Original results ranking is not affected (i.e., they are listed in the order they are returned by Amazon). However, the reasoner checks how many preferences a book does match and this is expressed by a star-rating system: The more stars that appear next to each title, the higher the book ranks in user preferences (Figure 6).

Furthermore, each item of the available offers list is marked with a different color depending on the offer's price. If it exceeds the user's preferable maximum price, it is highlighted in red. This allows the user to avoid ordering the book through that specific offer.

Additionally, the user can explore the full description of a book by clicking on its title. An "RDF Link" exposes the book information as Linked

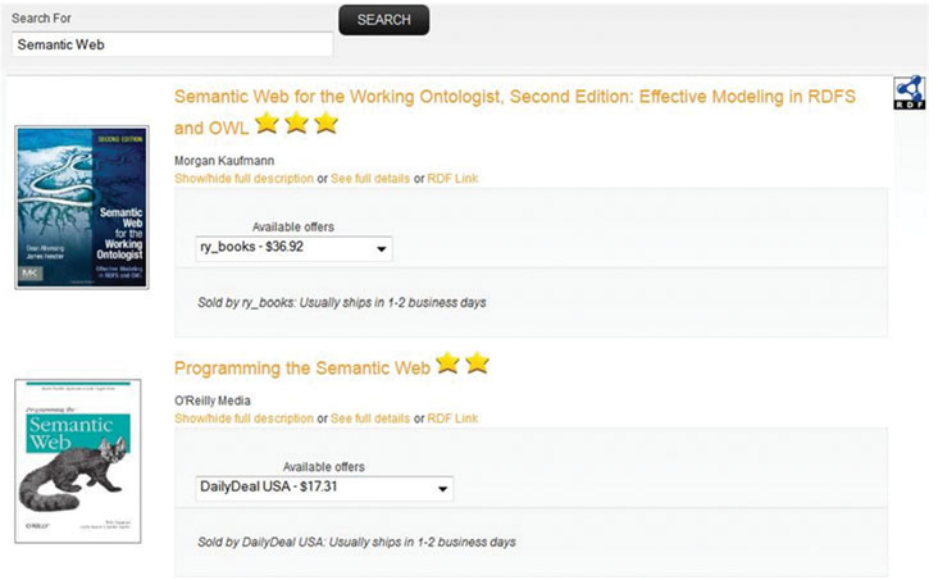

FIGURE 6 Result list and preference ranking (stars).

Data (in RDF format). Books' and authors' links are resolvable within our application and are interlinked with corresponding URIs of other applications (DBpedia). Finally, the search results lists are also available in RDF along with the inferred book rankings via the RDF icon button at the top of the books list.

Applications

Any data set published according to the Linked Data principles can be considered as an ideal data source for the development of a semantic mashup. Therefore, bibliographic information, expressed in either BIBFRAME or FRBR models, could flesh out the content of our mashup with well-documented bibliographic descriptions. On the other hand, the proposed deployment can help librarians and curators gather and assess available reviews.

There are many reviews—at Goodreads, Amazon, the *New York Times*, National Public Radio, and various other Web sites. Book readers are usually interested in gathering and viewing a list of reviews in total, rather than searching for a review of the book individually. In BIBFRAME 2.0, the class *Review* facilitates collection and evaluation in parallel of such reviews from the Web. Therefore, such instances of this class could be a valuable asset for our mashup. Besides, the integration of such sources can underpin our application's impact with the validity of the library community. In this way

the commercial world could come closer to libraries and open educational resources (OERs) (Johannesen, 2006).

Taking this idea a little further, our mashup can be augmented or even dedicated to reviews, instead of book records only. As such it can operate as a Linked Data container that associates readings, reviews, and the authority provided by library curation services. This mashup of information can then offer personalized fetching and ranking of reviews from corresponding providers.

Among the main features of the proposed mashup is its personalized search capability. Leaving out the whole Web-engineering view (CMS infrastructure), the proposed reasoning and Linked Data infrastructure can be applied in the context of a digital library, providing the benefits of a personalized search engine. The concepts of *User* and *Modality*, captured by the corresponding classes of the BookShop ontology, along with the power of reasoning services, ensure high search efficiency and effectiveness, as we will ascertain in the next section.

EVALUATION

We measure the performance of our semantic mashup and evaluate it based on *effectiveness* (how well does it match user search needs) and *efficiency* (how quickly are results retrieved). We present the evaluation procedure, which comprises a set of experiments within our specific evaluation framework, as well as the methodology of the experiments and their configuration. Finally, we discuss the results for each of the two evaluation dimensions.

Procedure

A sample of 10 users participated in the experiments and created manually their profiles. Each participant is required to issue 10 test key phrases in the search field of the application, which include the most frequent searches within Amazon (Table 4).Remember that relevance of results at this stage is determined solely by the data providers' algorithms. After the end of the search task, participants browse through search results and click on those books that presumably contain information that match their profile. The total

TABLE 4 Most Frequently Used Search Keywords in Amazon

Q1: big little lies	**Q6**: maze runner
Q2: cookbooks	**Q7**: stephen king
Q3: essential oils	**Q8**: the hobbit
Q4: fifty shades of grey	**Q9**: vegan
Q5: game of thrones	**Q10**: x-men

set of user's interactions (book item view, navigating through results' pages) with the application for a test keyword can be called *searching session*.

Metrics

In the context of the current evaluation, the effectiveness of our semantic search mashup is expressed via the *personalized search accuracy*. To estimate this, a modified version of the average rank metric (Qiu & Cho, 2006; Speretta & Gauch, 2005) is used for its measurement. The average rank of a keyword *s* taking into account all users' actions is defined as below:

$$AvgRank_s = \frac{\sum_{p \in P_{s_1}} R(p) + \sum_{p \in P_{s_2}} R(p) + \cdots + \sum_{p \in P_{s_{10}}} R(p)}{|P_{s_1}| + |P_{s_2}| + \cdots + |P_{s_{10}}|}$$

Here, P_{s_i} denotes the number of book items that user *i* has clicked for the test keyword *s*, while *R(p)* denotes the rank of book *p*. In this version, *R(p)* can take any of the following distinct values in C = {1, 2, 3, 4}, since books are ranked depending on the number of user preference criteria that are satisfied. For example, if user #1 has clicked the book *p* that satisfies only three preference criteria, then the value of *R*(*p*) is 3. The final average rank on test keyword set *S* is computed using the following equation:

$$AvgRank = \frac{1}{|S|} \sum_{s \in S} \mathrm{AvgRank_s}$$

In our case, the value of average rank indicates better placements of relevant results or better result quality when it is closest to 4. In order to keep it independent from the number of criteria, we can also normalize this value, as shown below:

$$AvgRank_{norm} = \frac{AvgRank}{|C|}$$

Continuing with the evaluation of the efficiency of our application, we measure the *response time* for a given query. The response time is considered as the total of two components, *retrieval time* and *reasoning time*. With retrieval time, we mean how long it takes to fetch and triplify the mashed-up data coming from the external Web sources. The different aspects of the retrieval time are consistent with the main subfunctions of the synchronization algorithm for the data storage (see Table 1, `get_amazon_data`, `get_ebay_data` and `triplify`).

Next, the reasoning time conveys the internal time that is needed for the reasoner to classify the ontology and evaluate the personalization rules against the data set. It also includes the time for the exchange of REST

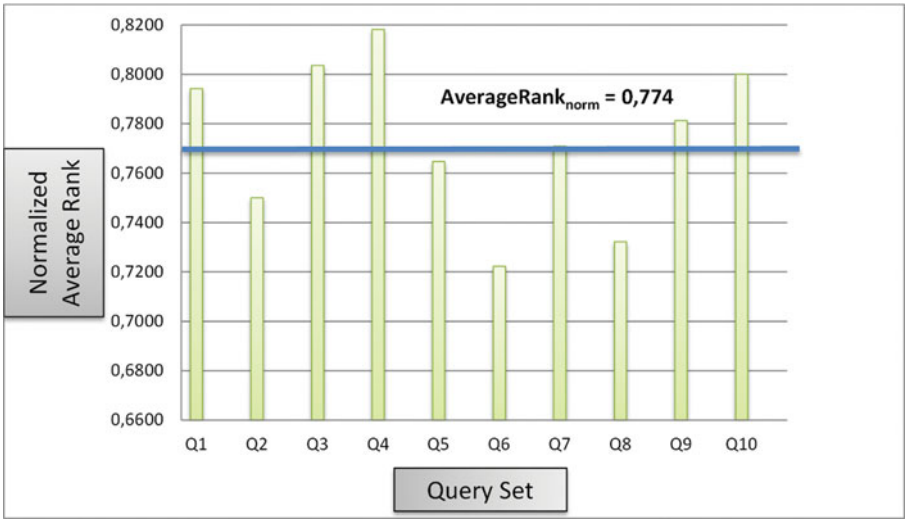

FIGURE 7 Normalized average rank for the experimental data set.

messages between the semantic mashup and reasoner over HTTP. In reference to the synchronization algorithm, the reasoning time corresponds to the `invoke_reasoner` function.

Results

Figure 7 depicts the normalized average rank for each of the 10 test queries, as well as the overall normalized *AvgRank*. We notice an $AvgRank_{norm}$ of 0.774, which means that, on average, relevant results are within the highest (4 criteria) or second-to-highest (3 criteria) ranking groups. In turn, this suggests that the ontology-based profiles and rules are able to deliver personalized search results that stand up well to user preferences. To have an appreciation of the meaning of such ranking scores, independent studies for Bing and Google report a reciprocal rank of 0.898 and 0.932, respectively (Liu, 2012), or even lower (Dou, Song, & Wen, 2007; Kumar, Park & Kang, 2008).

TABLE 5 Retrieval and Reasoning Times

	get_amazon_data (sec)	get_ebay_data (sec)	triplify (sec)	invoke_reasoner (sec)	Total (sec)
Avg. per session	1,744	5,847	0.021	1,717	9,328
Unit Cost	2,004	7,251	0.028	1,838	11,122

Table 5 summarizes the distinct parts of retrieval time, reasoning time, and total response time for our whole experiment. In particular, the amortized times per searching session are compared to the corresponding times of the unit cost. By unit cost, we mean the time needed for a new user to search a keyword viewing only one page of results and without mashed-up data being cached. It could be supposed that this is the maximum time for a keyword search.

We notice a difference for the retrieval times and not for the reasoning time. During our experiment, some searching sessions have been accomplished by using only cached data. So, for these sessions, the *get_amazon_data* and *get_ebay_data* aspects were negligible. The high retrieval times are due to the increased number of requests made to the Web APIs: Amazon and Half.com only allow fetching their results across many pages, which forces us to make multiple successive requests.

Although the algorithm implements a reasoner cache, amortized reasoning times are not significantly lower than the unit cost. This is due to the fact that this cache is useful only when there are repeating queries under the same user profile—that is, reasoning results depend on both the query *and* the user profile. Since the experiments did not involve repeating queries, the reasoning cache effect is unnoticeable.

CONCLUSIONS

Integration of Semantic Web applications with a CMS is not always straightforward. In order to achieve a seamless alignment, a series of issues has first to be resolved, and in this paper we have indicated exactly how this can be achieved in the case of our semantic mashup. Primarily, the semantic-oblivious nature of most CMSs calls for the explicit manipulation of semantically enriched data, which can be far from trivial, especially when their robust relational back-end is to be taken advantage of. Additionally, incorporating a reasoning infrastructure needs to be carefully designed as there may be substantive trade-offs involved.

Nevertheless, by combining the best of both worlds, the developer can genuinely focus on the internals of the Semantic Web implementation and assign Web content management and delivery on tried-and-true existing frameworks, instead of wasting time and effort. It turns out that, by investing in this integration, even the semantic aspects can benefit from data caching or reasoner delegation, thus making a virtue out of necessity. In addition, the CMS infrastructure can be inexpensively utilized in order to align our ontological data with the Linked Data principles, associate them with additional resources, and make them available to the LOD cloud. Moreover, the enrichment of our semantic mashup with REST features strengthens the reusability of semantic-enhanced data over the Web.

Beyond the Web-engineering aspect of our application, in this paper we have shown how commercial metadata can be enriched with bibliographic information by using Linked Data principles. In order to increase the interoperability of the gathered information from online bookstores with digital libraries' resources, we designed our underlying ontology by considering the specification of FRBR and BIBFRAME bibliographic models. This liaison gives new perspectives on both sides, by increasing the validity of the information offered by our application and adding the capability of a personalized search engine to existing digital libraries.

NOTES

1. http://linkeddata.org/
2. Available at http://swig.hpclab.ceid.upatras.gr/SWIGroupPapers/BookShop
3. http://www.loc.gov/bibframe/docs/bibframe2-model.html
4. http://www.rdaregistry.info
5. http://www.loc.gov/bibframe/docs/

REFERENCES

Antoniou, G., Baldoni, M., Baroglio, C., Baumgartner, R., Bry, F., Henze, N., May, W., Patti, V., & Wierzchon, S. T. (2005). Personalization for the Semantic Web II. Technical Report IST506779/Hannover/A3-D4/D/PU/b1, *Reasoning on the Web with Rules and Semantics*, August 31st, REWERSE.

Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., & Sheets, D. (2006). Tabulator: Exploring and analyzing linked data on the semantic web. Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI06), 6 November, Athens, Georgia.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American Magazine*, *284*(5), 29–37. Retrieved from http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF2

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked Data—The story so far. Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS) *5*(3), 1–22.

Bratsas, C., Bamidis, P., Dimou, A., Antoniou, I., & Ioannidis, L. (2012). Semantic CMS and Wikis as Platforms for Linked Learning. In Proceedings of the 2nd Int. Workshop on Learning and Education with the Web of Data (LiLe-2012 at 24th WWW-2012), April, Lyon, France. *CEUR workshop proceedings*, *840*.

Clark, L. (2011). The Semantic Web, Linked Data, and Drupal Part 1: Expose your data using RDF (Techical report). *IBM developerWorks*. Retrieved from http://www.ibm.com/developerworks/library/wa-rdf/

Corlosquet, S., Cyganiak, R., Decker, S., Polleres, A. (2009) Semantic Web Publishing with Drupal. Technical report DERI (Digital Enterprise Research Institute-TR-2009-04-30), DERI. Retrieved from http://www.deri.ie/fileadmin/documents/DERI-TR-2009-04-30.pdf

Davis, I., D'Arcus, B., & Newman, R. (2005). Expression of core FRBR concepts in RDF. Retrieved from http://vocab.org/frbr/core.html

Dou, Z., Song, R. & Wen, J.R. (2007). A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th international conference on World Wide Web (WWW '07)*, May. ACM, New York, NY, USA, 581–590.

Huajun, C., Bin, L., Yuan, N., Guotong, X., Chunying, Z., Jinhua, M., & Zhaohui, W. (2009). Mashup by surfing a Web of data APIs. *Proceedings of the 35th International Conference on Very Large DataBases* (VLDB 2009), August, Lyon, France *2*(1), 538–549.

IEEE LTSC. (2008). Draft recommended practice for expressing IEEE learning object metadata instances using the Dublin Core Abstract Model (IEEE P1484.12.4/D1).

Johannesen, Ø. (2006). Open Educational Resources: Policy Implications, presentation at Barcelona OECD Expert Meeting, October 2006. Retrieved from OECD, http://www.oecd.org/edu/ceri/37667317.pdf.

Kalou, A., & Koutsomitropoulos, D. (2015). Towards semantic mashups: Tools, methodologies, and state of the art. *International Journal of Information Retrieval Research*, *5*(2), 1–25.

Kalou, K., Pomonis, T., Koutsomitropoulos, D., & Papatheodorou, T.S. (2010). Intelligent Book Mashup: Using Semantic Web Ontologies and Rules for User Personalisation. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on, Int. Workshop on Semantic Web and Reasoning for Cultural Heritage and Digital Libraries (SWARCH-DL 2010)*, September 2010, (pp. 536–541).

Koschmider, A., Torres, V. & Pelechano, V. (2009). Elucidating the mashup hype: definition, challenges, methodical guide and tools for mashups. In *Proceedings of the 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009)*, 18th International World Wide Web Conference (WWW 2009), April, 1–9.

Koutsomitropoulos, D., Solomou, G., Pomonis, T., Aggelopoulos, P., & Papatheodorou, T. S. (2010). Developing distributed reasoning-based applications for the Semantic Web. In *Proceedings of the IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Perth, Australia, April 2010, 593–598.

Kumar, H., Park, S., & Kang, S. (2008). A Personalized URL Re-ranking Methodology Using User's Browsing Behavior. In *Proceedings of the 2nd KES International conference on Agent and multi-agent systems: technologies and applications (KES-AMSTA'08)*, March 2008, Springer-Verlag, Berlin, Heidelberg, 212–221.

Liebig, T., Luther, M., Noppens, O., & Wessel, M. (2011). OWLlink. *Semantic Web*, *2*(1), 23–32.

Liu, Y. (2012). *Evaluation Report for COS435 Retrieval, Discovery and Delivery Assignment 2*. Retrieved from Department of Science, University of Princeton website: http://www.cs.princeton.edu/courses/archive/spring12/cos435/Probs/ps2_report.html

Manguinhas, H., Freire, N., Machado, J., & Borbinha, J. (2012). Supporting multilingual bibliographic resource discovery with functional requirements for bibliographic records. *Semantic Web*, *3*(1), 3–21.

Manola, F., & Miller, E. (2004). Resource Description Framework (RDF) concepts and abstract syntax. W3C recommendation. *W3C*. Retrieved from http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

Ngonga Ngomo, A.-C., Heino, N., Lyko, K., Speck, R., & Kaltenböck, M. (2011). SCMS—Semantifying content management systems. In *Proceedings of the 10th International Semantic Web Cnference (ISWC)*. Bonn, Germany, October 23–27, Part II, 189–204.

Noppens, O., Luther, M., & Liebig, T. (2010). The OWLlink API-teaching OWL components a common protocol. In the *Proceedings of the 7th Workshop on OWL: Experiences and Directions,* June 2010. CEUR workshop proceedings, *614*.

Patel, S. K., Rathod, V. R., Prajapati, J. B. (2011). Performance analysis of content management systems-Joomla, Drupal and WordPress. *International Journal of Computer Applications*, *21*(4), 39–43.

Qiu, F., & Cho, J. (2006). Automatic identification of user interest for personalized search. In *Proceedings of the 15th International Conference on World Wide Web (WWW '06)*, May 2006, Edinburgh, Scotland, UK, (pp. 727–736).

Solomou, G., Kalou, K., Koutsomitropoulos, D., & Papatheodorou, T. S. (2011). A mashup personalization service based on Semantic Web rules and Linked Data. In *Proceedings of the 2011 Seventh International Conference on Signal Image Technology & Internet-Based Systems (SITIS '11)*, November, IEEE Computer Society, Washington, DC, USA, (pp. 89–96).

Speretta, M., & Gauch, S. (2005). Personalized search based on user search histories. In the *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI '05)*, September, IEEE Computer Society, Washington, DC, USA, (pp. 622–628).

Takhirov, N., Aalberg, T., Duchateau, F., & Žumer, M. (2012). FRBR-ML: A FRBR-based framework for semantic interoperability. *Semantic Web*, *3*(1), 23–43.

Tomlinson, T. (2010). Beginning Drupal, 7. (1st ed.). Apress, Berkely, CA, USA.