

# Towards Semantic Mashups: Tools, Methodologies, and State of the Art

*Aikaterini K. Kalou, Department of Computer Engineering and Informatics, University of Patras, Patras, Greece*

*Dimitrios A. Koutsomitropoulos, Department of Computer Engineering and Informatics, University of Patras, Patras, Greece*

---

## ABSTRACT

*Semantic Mashups constitute a relatively new genre of applications that illustrate the combination of the current trends of the Web, i.e. the Semantic Web and Web 2.0. The great benefit of Semantic mashups lies in the ability to aggregate different and heterogeneous data with rich semantic annotations and due to this, an additional ease of integration. In this paper, the authors attempt to outline the transition from conventional to semantic mashups, analyzing the former's limitations and identifying improvements and contributions which can come in with the advent of the later. Furthermore, the authors survey the background technologies on which semantic mashups are based, like Semantic Web Services and the process of data triplification. The authors also investigate the current trends and efforts put into developing tools and frameworks, which are designed to support users with little programming knowledge in semantic mashup application development, such as Semantic Pipes or Jigs4OWL. After presenting and illustrating the theoretical and technological background of this genre of mashups, the authors look into some use cases and systems. Among others, the authors present their mashup, called Books@HPClab, in which they introduce a personalized semantic service for mashing up information from different on-line bookstores.*

**Keywords:** *Automatic Tools, Conventional Mashups, Data Triplification, Linked Data, Manual Development, Ontologies, Programming Techniques, Semantic Mashups, Semantic Web Services, Semantic Web, Use Cases, Web Services*

---

## INTRODUCTION

The Semantic Web (Berners-Lee, Hendler, & Lassila, 2001), according to the inspirator of the Web, Tim Berners-Lee, is an extension of the current Web, in which data are more easily machine-understandable, which means that machines can better process, “understand” and integrate the information that they simply display at present. In recent years, with the evolution of

the Web 2.0, the maturity and the progress of APIs was a great opportunity for new genres of web applications, such as mashups, to emerge. A mashup is a Web-based application that is created by combining and processing on-line third party resources e.g. APIs, Linked Data, etc. This kind of combination is not limited only to data but also to presentation or functionality.

A valuable approach of combining these two trends of the Web (Semantic Web and

DOI: 10.4018/IJIRR.2015040101

Web 2.0) is the enrichment of mashups with semantics. This kind of web applications is usually called semantic mashups. In the context of Web 2.0, users are usually characterized as consumers or producers, due to the fact that they can consume offered data so as to reuse/refactor them and produce new kinds of data or a novel application. Therefore, semantic mashups can offer users-consumers a wealth of different and heterogeneous data with rich semantic annotations which thus offering an additional ease of integration. The great proliferation of semantically enabled web services makes the development of semantic mashups an easier and straightforward process. Due to the large amount of available information on the Web, the need for personalization is more intensive than ever for Web search. So, the potential of Web personalization ensures that each user is considered as a unique entity taking into account his different information needs and tastes. Note also that personalized data is delivered more and more via Web, making this kind of personalization viable and effective. Since the idea of personalization is embedded within the very nature of the Semantic Web, semantic mashups can become paradigms upon which personalization services can be delivered more efficiently.

The general objective of this paper is to analyze the main characteristics of a semantic mashup and to give an introduction to the progress of mashups. In Section *Towards Semantic Mashups*, we refer to several drawbacks and limitations of conventional mashups that led to the development of semantic mashups. A set of corresponding requirements and specifications for the latter is then then identified and discussed. In Section *Background Technologies*, we survey the background technologies on which semantic mashups are based. Next, we present several approaches for semantic mashup development, including architectures, languages and design frameworks. Next, in Section *Semantic Mashup Tools*, we investigate the current trends and efforts put into developing tools designed to support users with little programming knowledge in semantic mashup ap-

plication development, such as Semantic Pipes or Jigs4OWL. After presenting and illustrating the theoretical and technological background of this genre of mashups, we look into some use cases and systems in several domains of interest such as tourism or e-commerce. Among others, we present our mashup, called Books@HPClab, in which we introduce a semantic service for mashing up information from different on-line bookstores, producing personalized book recommendations and integrating them into the Linked Open Data (LOD) cloud.

## TOWARDS SEMANTIC MASHUPS

Despite the efforts to alleviate the challenging and time consuming task of mashup construction, many limitations still obstruct mashup developers. Among the main drawbacks, that mashup developers have to overcome, are service interoperability, reuse, integration and mediation. Nevertheless, Semantic Web technologies seem to address these limitations. In this section, we make an effort to mention the limitations for conventional mashups. Driven by the need to overcome these obstacles, we then set the stage for the notion of semantic mashups and attempt to deliver a preliminary definition and specifications.

### Limitation of Conventional Mashups

The most important limitations of conventional mashups are often inherent in the traditional mashup development approaches. Roughly, one can divide between two approaches, the *manual development* (programming or scripting) and the implementation of (semi-) *automatic tools*.

Regarding the manual development approach, potential problems may relate to one or more of the possible manipulation phases (conversion, filtering, format transformation, combination etc.) which actually form the data mediation component of a mashup application. All these data manipulations require mashup authors to be experts in the fields of

web technologies and to have a high level of programming skills (Tuchinda, Szekely, & Knoblock, 2008; Wong & Hong, 2007). Even more experienced developers confront many difficulties to transfer the problem domain representation into a detailed, and technical language of the programming or scripting environment (Schwarzkopf, Bauer, & Dengler, 2003). The problem is always the same, i.e. unstructured data needs to be processed in order to extract meaning and create structured data. Therefore, it is difficult either for novice or power users to create personalized mashup applications in a reasonable time-scale.

In order to solve this problem, many organizations have done attempts to develop (semi-) automatic tools for this purpose. Such tools come to cover the creation of mashups by end-users and limit the necessity for demanding skills and technical knowledge. However, they still require detailed background knowledge from the side of mashup designers. In all (semi-) automatic tools, the end-user has to select manually the heterogeneous and multiple data sources that should be remixed and aggregated in the mashup application. Manual data source selection does not only require the location (e.g. URL) of the data sources, but also knowledge about the structural and semantic diversities of the information that could be retrieved and combined. This structural diversity can rise problems like: (1) the same element, in two different schemas is given different names (e.g. latitude or geo:lat) or use two different data types (e.g. integer or double); (2) different elements are given same names but different meaning; (3) same elements in two different schemas are grouped in different ways; (4) one schema can cover some aspect of the domain that is not considered in the other schemas (domain definition).

Both manual and semi-automatic approaches of mashup creation also lack characteristics such as adaptivity, extensibility and interoperability. This means that if the structure of the provided data and the behavior of used functionalities are modified, the mashup application should be reengineered by its creators.

Additionally, if an end-user wanted to add a new service as data source in the existing mashup, it should modify its design in order to incorporate the new service's interface.

All the described limitations can be summarized in three distinct points:

1. Structural diversity of data sources,
2. Non-uniform semantics of aggregated data, and
3. Lack of adaptivity, extensibility and interoperability

Therefore, more automated and adaptive mashup authoring methods should be investigated and proposed. In the case of automatic generation, data sources are selected automatically and the retrieved data is combined without the need of human intervention. Furthermore, the ad-hoc creation of mashups avoids time consuming reengineering and automatically considers any changes. In order for such an automatic mashup generation to be achieved, an extensive utilization of machine processable and understandable semantics is required.

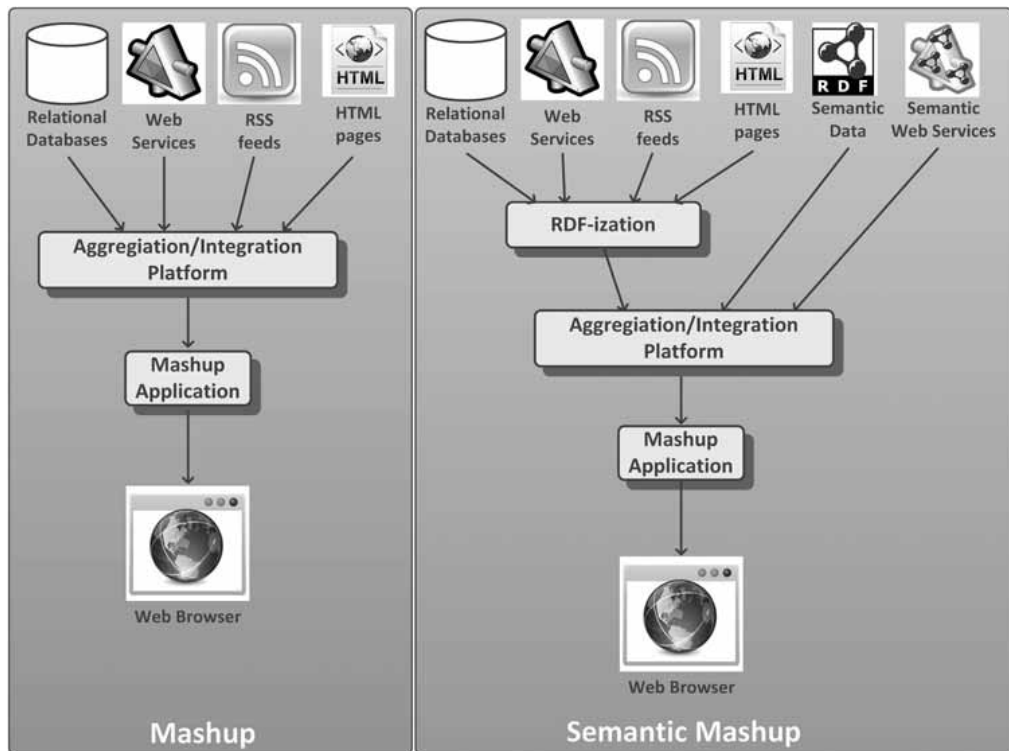
## Semantic Mashups Characteristics

These limitations are driving the development of the next generation of mashups - see Figure 1, called *semantic mashups*, relying on the already available semantic technologies. An early and intuitive definition of what can be considered a semantic mashup follows:

A semantic mashup constitutes a web mashup that employs semantic web technologies and ideas in any part of its design, architecture, functionality or presentation levels.

In accordance to the definition above a semantic mashup can be a mashup application that:

1. Aggregates data that are represented in an ontology schema, and consequently, expressed in any available semantic language, such as OWL;
2. Manipulates data expressed according to RDF specification, such as Linked data and

*Figure 1. Comparison between conventional mashups and semantic mashups*

retrieves those information using semantic technologies, such as SPARQL;

3. Consumes data from services which are enriched with semantics;
4. Handles data coming from RESTful services which are upgraded into semantically enabled services;
5. Arises from design techniques which involve ontologies in their implementation.

Specifications (1-2) imply that semantic mashups are able to address the challenges existing in the manual creation of conventional mashups and to resolve notably the problem of data integration and mediation. In particular, regarding the problem of manual mashup creation, the combination of semantic data relieves mashup creators of the load of unstructured data filtering and transformation. There is strong evidence that the integration and reuse of semantically annotated data is more straight-

forward than in the case of unstructured data, since methods such as ontology alignment (Noy & Musen, 2000), ontology merging (Noy & Musen, 2000; Noy & Musen, 2003) or ontology mapping (Ehrig & Staab, 2004; Noy & Musen, 2003) have been developed with desired results.

Ontology mapping methods can be used in order to represent correspondences between ontologies and can define how terms from one ontology can be related with terms from another. Next, ontology alignment (Shvaiko & Euzenat, 2013) is concerned with the (semi-) automatic discovery of correspondences between ontologies. Using ontology merging, a new, single ontology is created by unification (all the terms from the involved ontologies are included and mismatches between the overlapping ones are resolved) or by intersection (only the overlapping terms are included and their mismatches reconciliated). More and more, a great number of frameworks, that support

the usage of those techniques, emerge. See for example, the Alignment API framework (David, Euzenat, Scharffe, & Trojahn dos Santos, 2010) that can be exploited through a variety of ways such as command line, web interface and programming in order to obtain semantic data mediation via alignment.

The enhancement of Web services (main data source for a conventional mashup) with semantics was the aftermath of the proliferation of Semantic Web technologies and consolidates the service discovery, composition and execution. Therefore, the benefits of Semantic Web Services, when these are applied to mashups (requirements 3-4), seem to facilitate and simplify the manual selection of diverse data sources, which constitutes a significant barrier for (semi-) automatic mashup creation and limits their extensibility and adaptivity.

Finally, the exploitation of ontologies during the design phase of a mashup addresses efficiently the lack of adaptivity, extensibility and interoperability that is observed in any method of conventional mashup construction. These approaches achieve effectively the selection and composition of mashup components by using semantic technologies in the description of integrated data and in the discovery of data similarities.

The list of characteristics, that a semantic mashup should fulfill, corresponds to the set of semantic technologies, described thoroughly in the following section.

## BACKGROUND TECHNOLOGIES

In this section, we give an overview of the technologies that are facilitating the development of semantic mashups. Besides the “traditional” technologies for mashup creation, such as AJAX, SOAP/REST protocols, screen scraping and RSS/Atom, semantic mashup development can benefit of any technology that forms the Semantic Web backbone, such as ontologies, semantic representation languages (OWL, RDF), specifications for semantic data

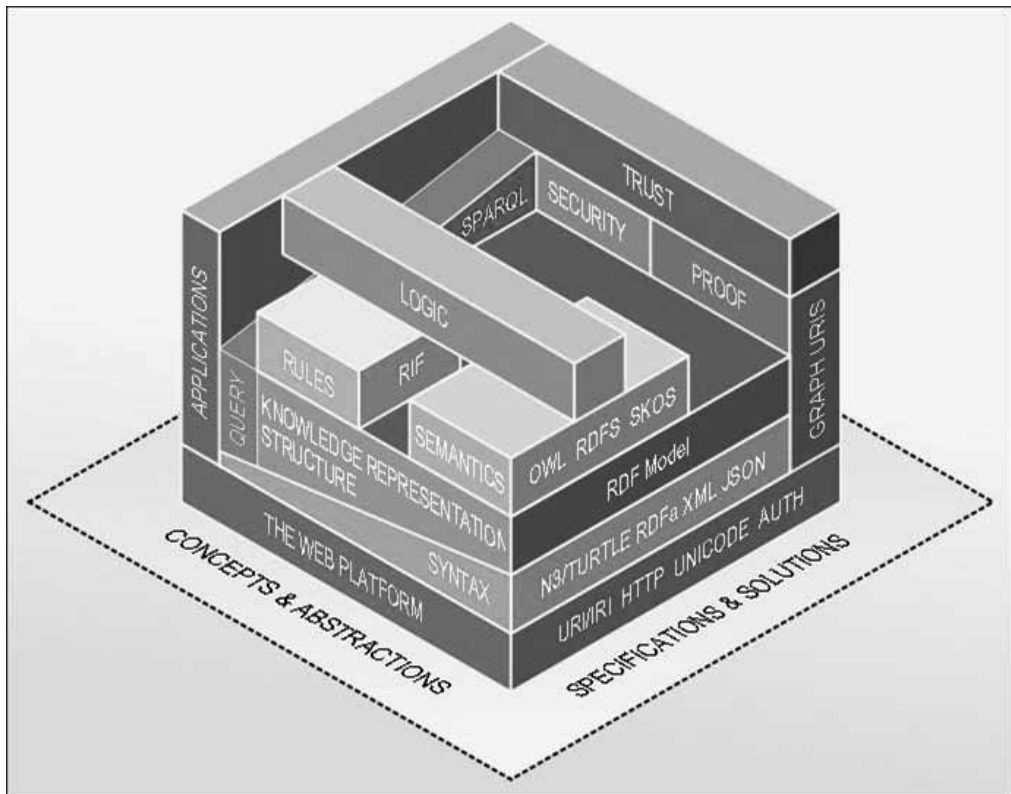
querying (SPARQL), specification for machine readable HTML pages (RDFa), and so on. Linked Data can serve as an umbrella over all these technologies and constitutes the particular pragmatic way of applying the technology stack of the Semantic Web.

## Semantic Web and Ontologies

The *Semantic Web* - see Figure. 2, 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. In this context, the Web content will be presented in a machine-understandable way, meaning that machines will become able not only to process and display, but also to “understand” and integrate available information (Berners-Lee et al., 2006).

*Ontology* is without doubt the “backbone” of the Semantic Web. The most simple and easily understandable definition of the term Ontology is proposed by W3C: “An ontology formally defines a common set of terms that are used to describe and represent the basic concepts in a domain and the relationships among them”. Since the Semantic Web relies heavily on formal ontologies, many languages for ontology expression such as SHOE, XOL, RDF, DAML and OWL and many tools for ontology engineering development have appeared, from WebOnto to Protégé. At the core of the Semantic Web architecture stack, as proposed by Tim Berners-Lee et al. (Berners-Lee et al., 2006), appears reasoning, the key component for the derivation of facts unexpressed explicitly in an ontology. Semantic reasoners and frameworks such as Pellet, FaCT++, Jena and others are pieces of software which implement this task. Ontologies are widely used in some use-cases analysed later in Section of *Use Cases* (Semantic mashup for Tourism and Books@HPCLab) as well as in some proposed design frameworks like SMASHAKER and the Mashup Rule Language

Figure 2. Semantic web stack



(Section *Methods and Techniques for Semantic Mashup Development*).

Among the semantic languages mentioned above *RDF* (Resource Description Framework) constitutes the most basic ontology language (Manola & Miller, 2004). More precisely, *RDF* specification is a language used for representing information about resources on the web. Resources are described in terms of properties and property values using *RDF* statements which are represented as triples, consisting of a subject, predicate and an object. The subject of one statement may be the object of another statement. A set of linked statements (triples) forms an *RDF* Graph. *RDF* Schema (*RDFS*) ‘semantically extends’ *RDF* to enable us to talk about classes of resources, and the properties that will be used with them. Although note that as expressive as *OWL*, the simplicity of *RDF*

makes it popular for implementing various aspects of semantic mashups. A representative sample of semantic mashup, which is totally expressed in *RDF* format, is the “*RDF* Book Mashup” (Bizer, Cyganiak, & Gauss, 2007). Figure 3 depicts an excerpt of *RDF* description derived from the deployment of *RDF* Book mashup.

*SPARQL* (Prud’hommeaux & Seaborne, 2008) is the de facto query language for *RDF* and since 2008 (v1.0) is a W3C recommendation. Similarly to *SQL*, which is used to query relational databases, *SPARQL* is a more sophisticated *SQL*-like language that intends to query *RDF* graphs. Currently, it is at version 1.1, a significantly enhanced version of its predecessor that comes with additional features for retrieving and manipulating *RDF* data. This kind of query language is widely used for specifying mashup

Figure 3. An excerpt of RDF description from the RDF Book Mashup

```

<foaf:Person rdf:about="/bookmashup/persons/Tim+Berners-Lee">
  <owl:sameAs rdf:resource=
"http://www4.wiwiiss.fu-berlin.de/dblp/resource/person/100007"/>
  <foaf:name>Tim Berners-Lee</foaf:name>
</foaf:Person>
<scom:Book rdf:about="/bookmashup/books/006251587X">
  <dc:creator rdf:resource="/bookmashup/persons/Tim+Berners-Lee"/>
  <rdfs:label>Weaving the Web</rdfs:label>
</scom:Book>
<scom:Book rdf:about="/bookmashup/books/8432310409">
  <dc:creator rdf:resource="/bookmashup/persons/Tim+Berners-Lee"/>
  <rdfs:label>
    Tejiendo La Red - El Inventor del WWW Nos Descubre
  </rdfs:label>
</scom:Book>

```

languages (see sub-Section *MashQL*), as well as for tools for semantic mashup development (see sub-Section *Semantic Web Pipes*). Besides, a set of interesting SPARQL extensions, such as SPARQL+ and SPARQLScript, proposed in (Nowack, 2008), appends a complementary group of tools for semantic mashup creation. Figure 4 indicates the extension of SPARQL with the AGGREGATION operation.

All these Semantic Web technologies constitute an environment capable of enabling efficient and personalized applications, since the idea of personalization is embedded within the very nature of the Semantic Web (Antoniou et al., 2005).

## Linked Data and Semantic Annotations

In recent years, among the means to reach the vision and accomplish the scope of Semantic Web is also the *Linked Data* technology. In summary, the concept of Linked Data focuses on the creation of typed links between different data sources by using the Web. Linked Data, from the 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 from external data sets. The result of this connection is the Web of Linked Data. ‘Linked Data principles’, a set of best practices by Berners-Lee, define how to publish and connect data on the Web, so

Figure 4. SPARQL+ - A SPARQL extension

```

SELECT COUNT(?contact) AS ?contacts WHERE {
  <#me> foaf:knows ?contact .
}
ORDER BY DESC(?contacts)

```

that all published data becomes part of a single global data space (Bizer, Heath, & Berners-Lee, 2009). At present, the concept of Linked Data seems to be applicable in the context of Big Data by providing the potential of more effective data integration and management (Mitchell & Wilson, 2012).

Among several solutions for applying the semantic concepts to existing web applications in order to transform their content to machine readable content, the RDFa initiative of W3C (Adida, Birbeck, McCarron, & Pemberton, 2008) seems to be widespread for semantic mashup development. RDFa, which is an alternative syntax for RDF, makes the available data on the Web more intelligent by embedding semantics into them at the creation time. As it is obvious, the most interesting aspect of RDFa is that it can be embedded in any HTML page, so that it is easy to state what things on an HTML page actually mean. In RDFa, a set of HTML attributes are defined so as to be added as metadata in HTML tags and augment data with machine-readable hints. Therefore, the use of RDFa enables data to be easily shared and reused. RDFa can also facilitate operation of search engines, which can therefore deliver more relevant search results.

A similar approach for embedding machine-readable data in web content is also delivered by *microformats* (Khare & Çelik, 2006). Microformats have grown out of the work of the blog developer community as an easy and ad-hoc response to common applications, whereas RDFa is built with a more systematic vision of the W3 Semantic Web group.

Another succinct format for communicating linked data and their semantics is the so called JSON-LD (Sporny, Longley, Kellog, Lanthaler, & Lindstrom, 2014) which constitutes a lightweight syntax to serialize Linked Data in JSON. JSON-LD involves the notion of context in its syntax in order to enrich the represented data with provision information. Its design has a twofold character, able to support Semantic or non-Semantic technologies. On one side, its compatibility with existing JSON, gives the capability to integrate Linked Data

within web-based programming environments, to build interoperable Web Services as well as to store Linked Data in JSON-based storage engines (MongoDB). Systems already using JSON can easily be upgraded to JSON-LD. On the other side, a JSON-LD document can also be considered as an RDF document capable of cooperating with existing RDF tools according to JSON-LD extensions to the RDF data model (Cyganiak, Wood, Lanthaler, 2014). Given that no knowledge of RDF is a prerequisite in case of JSON-LD usage, it is an opportunity for Web developers, unfamiliar with the notion of Semantic Web, to enrich their work with semantics.

## METHODS AND TECHNIQUES FOR SEMANTIC MASHUPS DEVELOPMENT

In this section, several approaches for semantic mashup development are presented (see Table 1). Firstly, we begin with a fundamental technique, called *data triplification*, which can be considered as a background for many tools and semantic mashup applications. We continue by introducing the technology of Semantic Web Services, which constitutes the leader data source in order to develop a semantic mashup. Next, we focus our study on the part of appropriate mashup languages. Finally, we present design frameworks that are commonly used at a proactive stage of semantic mashup development and have recommendation features, aiming to support developers in the effective selection of mashup components. The specific applications and tools for creating semantic mashups out-of-the-box are discussed in the next section (Section *Semantic Mashup Tools*).

### Data Triplification

As mentioned above in section *Semantic Mashups Characteristics*, a mashup could be characterized as semantic either due to the fact that it consumes data expressed in powerful formalisms like RDF/OWL or because its design utilizes semantically enhanced data. Therefore,



Table 1. Summary of approaches for semantic mashup development

|                   | Name                  | Semantic Web Technologies    | Scope/Purpose  |
|-------------------|-----------------------|------------------------------|--|
| Techniques        | Data Triplification   | RDF<br>OWL<br>Linked Data    | –Data conversion to machine-readable data<br>–Supportive method for tools for semantic mashups<br>–Applicable to many semantic mashups   |
|                   | Semantic Web Services | RDFa<br>Microformats JSON-LD | – Description of services' capabilities and content in an unambiguous, computer-interpretable language<br>–Improvement of the quality and robustness of existing tasks<br>–Automated discovery, invocation, composition, monitoring and execution          |
| Mashup Languages  | MashQL                | RDF<br>SPARQL                | –Query and mash up web data sources easily<br>–No knowledge for the underlying structure or technical details of the data sources is required  |
|                   | Mashup Rule Language  | Ontology                     | –Appropriate for average users<br>–System for mashup of Web resources construction<br>–General or Adaptive mashup  |
| Design Frameworks | SMASHAKER             | Ontology<br>OWL              | –A recommendation system focused on (semi-)automatic support of mashup design<br>–Proactive and interactive mashup component ( <i>data source</i> ) selection and composition<br>–The system needs semantically enhanced data sources (SA-REST and SAWSDL) |
|                   | sMash                 | RDF                          | –A system for facilitating users to mashup Web data<br>–The systems helps novice users master data APIs and relationships amongst them easily<br>–The system inspires various users to build more engaging Web data mashups                                |

the steps' sequence for a more generalized method to turn any kind of data in machine-readable data should be formulated. This method is known as data triplification (Matthew, 2010; Nuzzolese, Gangemi, Presutti, & Ciancarini, 2010). This data *triplification* method is used in many tools for semantic mashups as well as in the case of Books@HPCLab mashup (Section *Use Cases*). It is worth mentioning that only IT-experts could apply this method, since high level programming skills are required.

**Step 1. Select Data Sources:** First the appropriate data sources have to be identified and selected in order to match the mashup's scope. To retrieve data from these sources one can rely on available Web Services/ Web APIs (identified as RESTful services) or use more obsolete methods such as web scraping.

**Step 2. Analyze the Structure of Data:** After the data sources selection, the structure of the available data should be captured and analyzed so that useful information is

discovered. In the case of Web Services/ Web APIs, the returned data encapsulate the notion of structure. For example, data returned by RESTful Web services usually adhere to some external XML schema that implies their knowledge structure. So, to examine and understand this structure is sufficient in this case. On the other hand, web sites or other data sources, which do not offer their data in a structured way, require finding out and shaping the structure of data on the data consumer's side.

**Step 3. Select Useful Data:** Usually, the outcome of the previous step gives a bulk of information that sometimes include superfluous details which may be outside of the intended scope of the mashup. So, a set of the important data should be elected depending on the mashup's orientation and purpose, e.g. Figure. 5.

**Step 4. Communication with Data Sources and Data Retrieval:** This step involves communication with the selected sources and data retrieval in any available format (XML, JSON etc). For example, suppose that one selects the Amazon Web Service as data source (a REST-based API). Communication with this service is done simply over well-known HTTP methods. Just like most RESTful services, Amazon APIs can have a set of parameters to restrict the effect of the HTTP methods on the resource. The set of selected data from Step 3 determines the set of parameters that is going to be used in a GET request. This kind of requests is encoded in a URL string and the services answer simple XML or JSON. In the case of web scraping, the data extraction from any web site is implemented by using techniques such as human copy-and-paste, HTTP programming, data mining algorithms, DOM parsing, HTML parsers, Web-scraping software and so on.

**Step 5. Knowledge Schema Construction and Data Adaptation:** The main goal of the overall method is the enrichment of data with semantics. To this end, the development of an ontology that can capture

and represent effectively the meaning of retrieved data is a usual policy. The process of ontology development is mainly guided by the analyzed structure of data from Step 2 and the selected data of Step 3. However, the development of a schema is not sufficient by itself. The adaptation of data to the fully-developed representation schema is necessary and can be accomplished by using current technologies such as XSLT, RDF parsers, RDF libraries, ontology alignment or mapping. For example, in the RDF Book Mashup (Bizer et al, 2007), the resulting XML responses from Amazon APIs are parsed using PHP's SimpleXML functions. Then, they are turned into an RDF model which is serialized to RDF/XML using the RDF API for PHP, called RAP.

**Step 6. Data as Linked Data:** As a final step, a best practice is to make the data resolvable and to provide them as Linked Data.

The data triplification method, described above, constitutes a workflow and a basic background for many tools and mashup applications, oriented towards the Semantic Web. The steps' sequence method is not obligatory, and some steps can be omitted. Finally, the method could have great effects on mashups' scenarios that are based on data replication.

### Semantic Web Services: Easier-to-Use Services for Mashups

The great proliferation of web services, regardless of their implementation, be it SOAP, RESTful or other, has assisted towards the evolution of mashups. Tools such as Yahoo Pipes or IBM Mashup Center, which have been available for facilitating the time consuming task of mashups development, limit their operation in a targeted number of services. This limitation in the number of services that mashup designers could "consume" constitutes a remarkable drawback for these tools (Lathem, Gomadam, & Sheth, 2007). The users of these tools restrict themselves to select only among services that are internal to the organization that the tool was

Figure 5. The third step of the data triplification process



developed from or services that have standard outputs. Therefore, services with non-standard types of inputs or outputs are excluded from the pool of available services for consumption, since they require significant modifications to the existing tooling in order to be embedded. The task of addition or removal of a service from a given mashup is also characterized challenging (Lathem et al., 2007).

In this context, a number of well-known approaches to annotate web services semantically are found in the literature. Among these approaches, OWL-S (Martin et al., 2005), WSMO (Roman et al., 2005), SAWSDL (Farrell & Lausen, 2007) and WSMO-Lite (Vitvar, Kopecký, Viskova, & Fensel, 2008) are specified to be applied in the case of WSDL services, based on SOA. On the other hand, the main purpose of MicroWSMO and SA-REST (Gomadani, Ranabahu, & Sheth, 2010; Lathem et al., 2007), is to enhance with semantics Web APIs and RESTful services by adapting existing ideas and formalisms. In this section, we focus on these two semantic description approaches suitable for RESTful services. For completeness, we first make a brief reference

to the SASWDL standard which is used as a basis for both approaches.

SAWSDL (Semantic Annotations for WSDL and XML Schema) (Farrell & Lausen, 2007) is a more lightweight description formalism than the other approaches, but not so expressive. However, SAWSDL approach is very promising because it enables adding semantic information directly on top of existing service descriptions, by linking service keywords to ontology elements. Finally, it is the only current official W3C recommendation for semantic Web services.

Currently, simple HTML pages are used to describe to users what a RESTful service does and how to invoke it. Roughly, this category of HTML pages is to human users what WSDL descriptors are for machines. So, these unstructured HTML pages are exclusively oriented towards human consumption and the processing of RESTful services descriptions by machines is restrictive. In order to solve this problem, the idea of adding semantics to WSDL descriptions, in the case of SOAP services, was adopted by many research groups. The semantic annotations are meant to be implemented on top of existing HTML descriptions.

The two main contributions in this direction are MicroWSMO and SA-REST. Both approaches conform with the principles of SAWSDL formalism so as to mark service properties within the HTML description and link them to semantics. More concretely, either MicroWSMO or SA-REST formalism captures semantic annotations for service inputs, outputs, operations, faults and also the type of the request. MicroWSMO relies on the hRESTs microformat (Kopecký, Vitvar, Fensel, & Gomadam, 2009) for describing the main aspects of a RESTful service and seems to be an extension of hRESTs. SA-REST, on the other hand, uses RDFa for marking service properties. SA-REST use-cases include faceted search, data mediation, smart mashups, semi-automatic text annotation and service oriented sensor networks (Gomadam et al., 2010).

As it is obvious, these two approaches adopt the same underlying principles, but use different implementation techniques. Furthermore, MicroWSMO adds semantics as means for Web service automation, while SA-REST enables tool support for service discovery and mashup composition (Maleshkova, 2009).

## Mashup Languages

The numerous attempts that have been made in order to obtain simplified processes for mashup construction have resulted in several languages apart from tools [Kitchin, Quark, Cook, & Misra, 2009; Maximilien Maximilien, Wilkinson, Desai, & Tai, 2007]. The idea of mashup languages has also been adapted for the case of semantic mashups. In this section, some prominent mashup languages related to the notion of semantics are presented.

- **MashQL:** The remarkable success of Yahoo Pipes (Jarrar & Dikaiakos, 2008, Le Phuoc, Polleres, Morbidoni, Hauswirth, & Tummarello, 2009) in regard to other mashup editors motivated Jarrar et al (Jarrar & Dikaiakos, 2008) to propose MashQL, an interactive query formulation language. The idea of combining different

data sources into mashups, in a graphical and user-friendly way without having to write code, as proposed by Yahoo Pipes application, was adopted by the inspirators of MashQL. So, the goal of MashQL is to enable non-experienced end-users to create their data mashups diagrammatically.

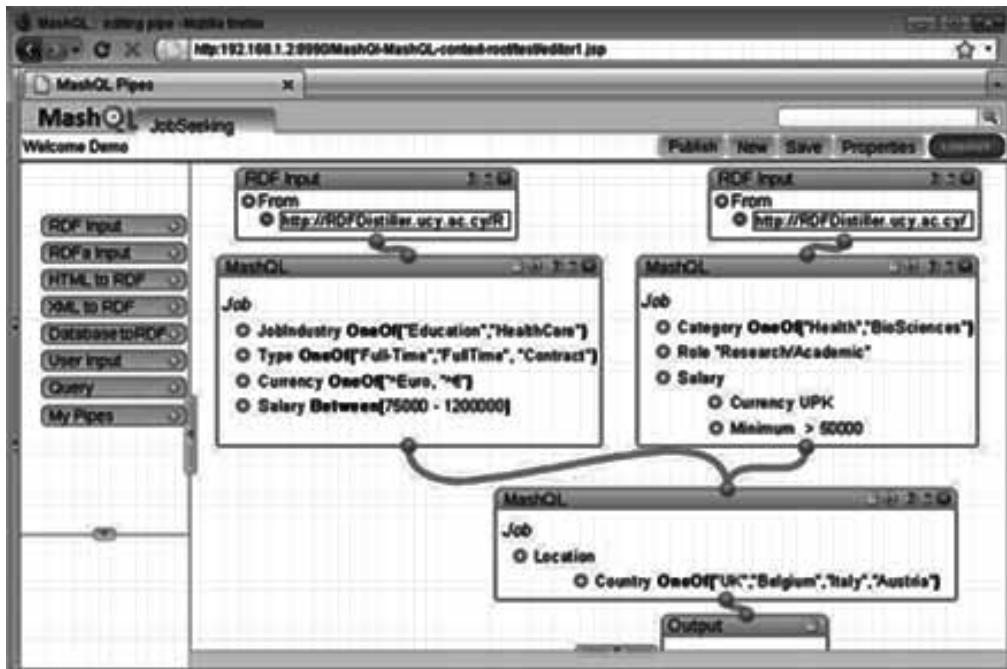
The main assumptions that have been made are that Internet is supposed to be a database and each internet data source is seen as a table and represented in RDF. Next, a mashup is considered as a query on these tables and the query is expressed by SPARQL. The novelty of MashQL is that no knowledge about RDF/SPARQL technologies nor any prior understanding of the schema or the structure of the consumed source are required in order to create the mashup.

Jarrar et al. (Jarrar & Dikaiakos, 2012) have proposed two implementations of MashQL, an online mashup editor - see Figure. 6- and a browser-side Firefox plug-in. These implementations are offered for free and can be downloaded at the language's website <http://sina.birzeit.edu/mashql/>. In order to execute the implementations, Firefox 3.5 or lower, and Java JRE or JDK are required.

MashQL functionality is comprised of three main modules, (i) the query input module, (ii) the query body module and (iii) the query output module. The first module specifies the data sources that are going to be queried. The second one specifies the query body. The query output is piped into the final module that renders the results into HTML or XML or as RDF input to other queries, converting the module in a new query input module. This last function allows pipelining of queries and reuse of data mashups as inputs to others mashups.

- **Mashup Rule Language:** In this subsection, we focus on the research work presented in (Jung & Park, 2011), which is about creating mashups using a mashup rule language. In brief, a system based on ontological models is proposed, aiming at supporting average users, that lack pro-

Figure 6. The online MashQL-Editor



programming experience in easily constructing mashups of Web sources. The Web sources that are going to be combined are the results of a search process implemented with the aid of the Google AJAX Search API. Users can start searching by simply setting a query and defining a mashup rule that specifies how the mashup is going to be constructed. Furthermore, an ontology restricting the scope of the search can be defined.

In any mashup rule, a user should firstly determine in which resources the system should search. Next, the layout of the results set should be specified. More precisely, the user denotes where on the mashup's web page obtained results are going to be displayed (in the center, at the right side, at the left side, at the top or at the bottom). In addition, he needs to define whether the search will be restricted to a specific web site or not and predicate any special query term. In the case of an adaptive,

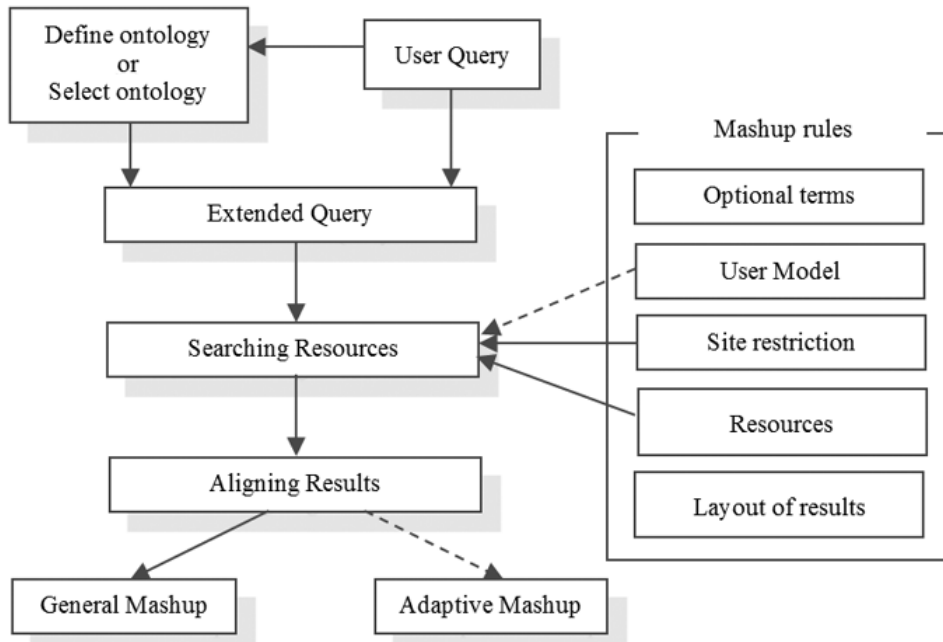
personalized mashup, a user model should be also determined in the body of the mashup rule.

After the mashup rule definition, it is necessary to decide whether an ontology will be taken into account in order to increase the system's effectiveness. In this case, either a new ontology about a specific domain should be specified or discover relevant ontologies from Swoogle (Ding et al., 2004) can be discovered. The overall system workflow and rule definition is shown in Figure. 7.

## Design Frameworks

Although the construction of mashups has been simplified to a large extent, the selection of mashup components must be done manually by mashup designers, a really complicated task. Additionally, the huge quantity of available mashup components presented in ProgrammableWeb.com, disorients the designers from selecting effectively appropriate components. Therefore efforts to provide support for effective selection of mashup components are gaining

Figure 7. Functionality of the mashup rule language system



more and more attention. These systems rely on the description and organization of mashup components and their proactive suggestion. In this section we present two such systems, SMASHMAKER and sMash.

- **SMASHAKER:** SMASHAKER (Semantic MASHup shAKER) (Bianchini, De Antonellis, & Melchiori, 2010; Melchiori, 2011) is a prototype tool environment that supports the designer in the proactive and interactive mashup component selection and composition. In brief, this framework collects mashup components from the Web, such as Web APIs and SOAP services. All these components are semantically described via proper wrappers, including SA-REST wrappers, RSS/Atom feeds or SAWSDL, depending on the data source. The semantic descriptors of components, given as XML documents, are stored in the Mashup Component Repository.

Semantic descriptors are organized according to the Mashup Ontology, in terms of similarity and coupling coefficients. The Similarity Evaluator and the Coupling Evaluator check the functional similarity and the degree of coupling between the semantic descriptors respectively, and rank the list of candidate components based on the designer's selection. All these functions are included in the Mashup Engine module of SMASHAKER.

By means of an available GUI, a mashup designer can specify a request for a desired component. Then the system replies to the user with a set of those components that present high similarity with the requested one. If one of the suggested components is selected from the designer, the initial similarity set is updated and additional components are suggested. These additional components may be similar to the one selected or can be coupled effectively with already selected ones during mashup construction.

- **sMash:** The concept of classifying Web APIs and their effective discovery and selection for mashup development seems to be also adopted in the case of sMash (semantic-based Mashup) (Bin et al., 2009; Huajun et al., 2009). The key idea behind this work is the construction and visualization of a real-life data API network which makes mashup building easier and inspires more interesting mashups via navigation. For each API, the system maintains a metadata description expressed in RDF format. In the API network, each API is represented as a node. A link between two nodes means that these two APIs can be combined, so they are linked via a 'mashupable' relationship. A user's navigation in this network defines a path/trace that can be conceived as a mashup. The system offers recommendation features to support both users that have clear ideas about how the mashup results should be like as well as users that attempt to construct a mashup with no specific purpose, by offering recommendation features. The recommendation task is accomplished by comprehensively analyzing the network of APIs, user's traces and a repository of mashups (Huajun et al., 2009).

Two kinds of recommendations are offered by the system, namely trace-based and inference-based recommendation, addressing novice users and experienced users respectively. In the case of trace-based recommendation, sMash analyzes the traces that occurred up to now and suggests a top-10 of the most popular links to go next. The inference-based recommendation relies on a repository that is formed by mashups listed in Yahoo Pipes and ProgrammableWeb. More precisely, a sample API network is constructed, including all the APIs that are used by the mashups of the repository. Next, sMash finds out the undiscovered APIs based on user's trace and visualizes them. Finally, the system infers and recommends the top-10 APIs by calculating the similarity

between trace nodes and nodes included in the repository (Huajun et al., 2009).

## SEMANTIC MASHUP TOOLS

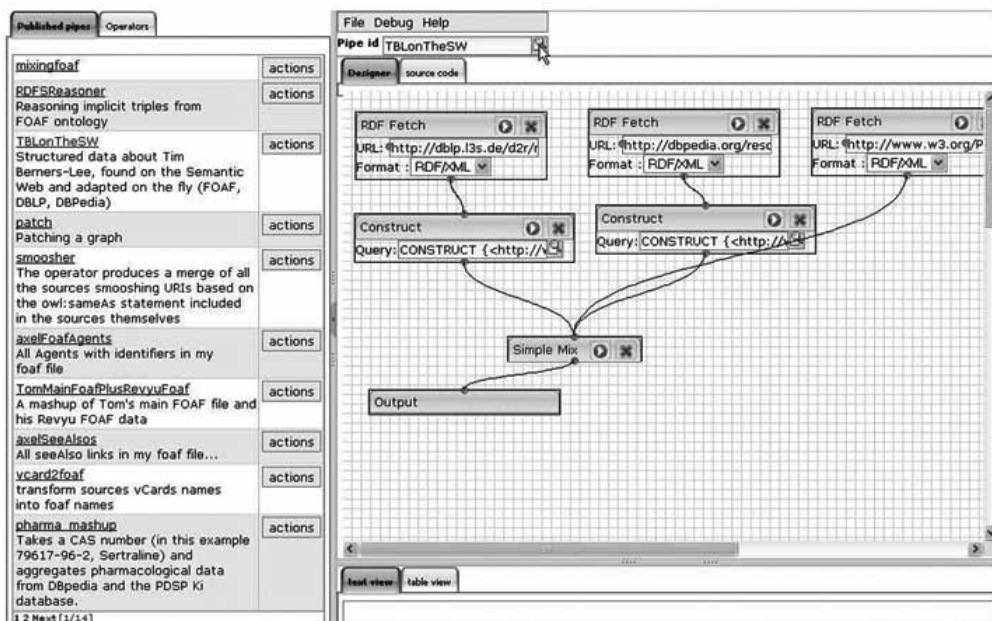
After presenting several approaches for deploying semantic mashups, we continue our survey with some tools that provide functionalities for building and publishing such mashups. The approaches, described in Section *Methods and Techniques for Semantic Mashup Development*, can only be exploited in combination for semantic mashup development. Contrary to the aforementioned approaches, the tools of this section could be considered as complete solutions for an end-to-end development of semantic mashups.

Although there is a rich literature on standards and approaches for semantic mashup development, the availability of mature, implemented, practical tools integrating these methodologies is limited. The most representative and fairly applicable tools in this direction include the following: *Semantic Web Pipes*, *Jigs4OWL* and *SWEET*. The first two facilitate the task of aggregating semantic data and utilize usual techniques such as pipelining and widgets correspondingly. The last tool facilitates mashup creation by supporting semantic annotation of RESTful Web services. A context-based comparative discussion about these tools is given at the end of this section, along with an overview of some other related efforts.

### Semantic Web Pipes

Semantic Web Pipes (SWP) or DERI Pipes (Morbidoni, Le Phuoc, Polleres, Samwald, & Tummarello, 2008; Le Phuoc et al., 2009) constitute a promising work based on the classical pipe abstraction. The latter has already been applied on parallel processing and has been adopted in Yahoo Web Pipes, by the most widely-used mashup tools. In short, SWP is both an engine and a GUI serving general Semantic Web Data transformations -see Figure. 8, as well as a powerful tool for building RDF-based mashups.

Figure 8. The GUI of Semantic Web Pipes



The RDF data aggregation, filtering and transformation tasks are performed by special purpose operators that can be arranged in the form of a pipe through a graphical Web editor. Therefore, a Semantic Web pipe is defined by a set of connected operators. The resulting pipe can in turn be used as an operator in other pipes. Supported operators range from RDF/XML extraction from existing Web content to basic SPARQL queries or more advanced application of RDFS/OWL inference rules. Every operator has at least one input and only one output. The output of a pipe can be an RDF model or an XML file, which can be retrieved via HTTP protocol. Likewise, a pipe or an operator has inputs which are string text, RDF, or XML. Finally, developers can extend the functionality offered by the proposed framework by creating new pipe operators.

SWP is an open-source implementation that contains a Web-based editor for easily composing pipes by drag&drop operators (the pipe editor). In addition, it has execution engine for supporting and executing these operators and a pipes repository for storing representa-

tions of pipes. SWP is available online at <http://pipes.deri.org:8080/pipes> via a Web interface. The execution engine of SWP allows pipes to be executed on the command line or to be embedded within an application, once the user downloads and unpacks the pipes-engine package from <http://sourceforge.net/projects/semanticwebpipe/files/>. Finally, users can develop and execute pipes from within the java applications via a powerful and simple API.

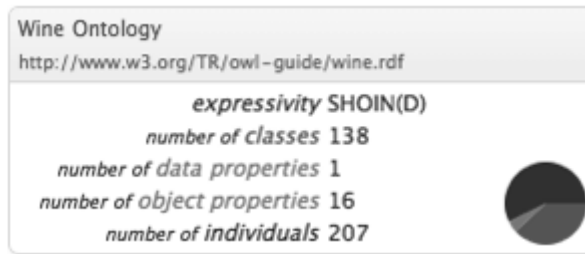
## Jigs4OWL

Jigs4OWL (<http://www.jigs4owl.com/>) constitutes a semantic mashup framework that allows the construction of web-based applications able to aggregate, visualize, and mashup semantic data. For facilitating the construction of such applications, Jigs4OWL offers several widget templates that deal with semantic content.

Widgets are snippets of code, capturing frequently used functionality, and executed usually within a web page. In the case of Jigs4OWL, each widget is a bundle of JavaScript files, typically hosted at the project's own



Figure 9. Info Widget: An example display for the Wine Ontology, whose expressivity is SHOIN(D)



premises, the Jigs4OWL server. Corresponding code is automatically fetched and the widget is instantiated and initialized on loading. As soon as the knowledge base is specified, the widget acquires access to the server's semantic infrastructure through its specific API methods.

At the moment, the functionality offered by all the Jigs4OWL's available widgets includes the following: a) faceted navigation of semantic data, b) graphical exploration of class hierarchies rendered as either trees or 2D maps, c) searching of knowledge bases via entity names, and d) information retrieval from a particular knowledge base, see Figure. 9. These widgets are customizable and can collaborate with other

third-party servers. Consequently, developers based on the Jigs4OWL framework for building their mashup applications can exploit provided widgets and combine them with other similar code modules or data sources on the web, without having to worry about ontology hosting, handling, accessing and reasoning issues.

In order to access semantic data, the Jigs4OWL server provides a SPARQL-DL endpoint. SPARQL-DL is a substantial subset of SPARQL that is used as a query language for OWL DL ontologies. The only types of SPARQL-DL queries supported in Jigs4OWL are ASK and SELECT. Finally, the Jigs4OWL framework can be either driven as an internal

Figure 10. The GUI of SWEET

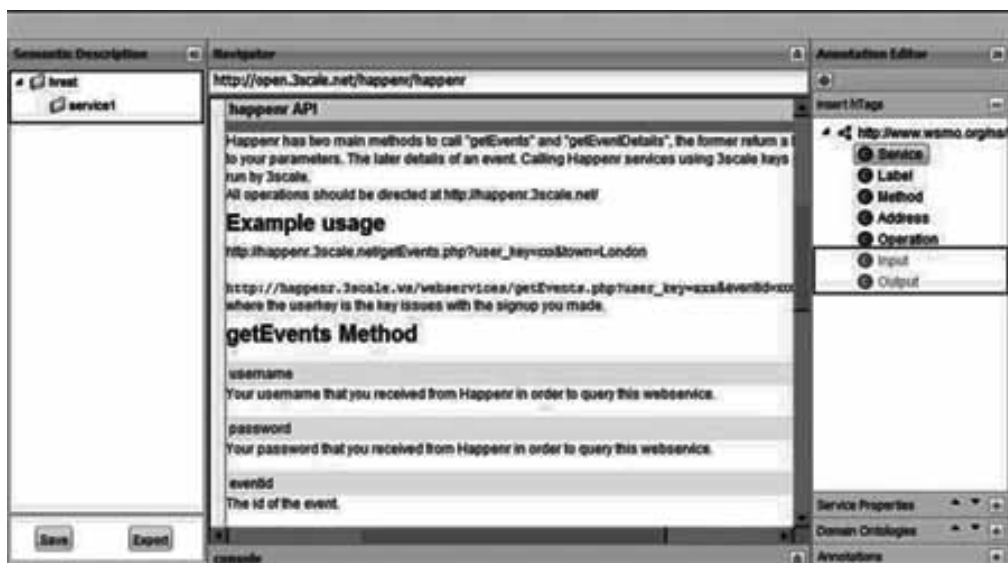


Table 2. Semantic standards and features supported by the semantic mashup tools

|                    | Semantic Technologies     | Functionality/Features  | Approaches for Semantic Mashup Development        |
|--------------------|---------------------------|---|---|
| Semantic Web Pipes | RDF<br>SPARQL<br>RDFS/OWL | – Semantic Web Data transformations<br>– Building RDF-based mashups<br>– Based on pipe abstraction  | Data Triplification<br>Initialization with MashQL |
| Jigs4OWL           | SPARQL<br>OWL-DL          | – Construction of web-based applications able to aggregate, visualize, and mashup semantic data<br>– Based on widgets<br>– Available SPARQL-DL endpoint | Data Triplification                               |
| SWEET              | RDF<br>Microformats       | – Creation of Semantic Mashups<br>– Semantic Annotation of RESTful Web Services   | Semantic Web Services                             |

or external service or employed on-demand from a framework service provider.

### SWEET: Semantic Web Services Editing Tool

SWEET [Maleshkova, Pedrinaci, & Domingue, 2010] is a Web application implemented using Javascript and ExtGWT, see Figure. 10. This tool provides support to users in semantically describing RESTful services by marking service properties, searching for suitable ontologies and attaching semantic information. The inspirators of this work use MicroWSMO formalism so as to achieve the semantic annotation of services.

In summary, SWEET takes as input an HTML page, which describes in detail the candidate RESTful service, and gives as output either an enhanced version of the HTML description, enriched with semantics, or an RDF MicroWSMO description. So, the user has to accomplish the following four steps Figure. 10:

- **Step 1:** Mark service properties by inserting hRESTs tags within the HTML page (Navigator panel).
- **Step 2:** Search among a set of domain ontologies in order to select the most suitable for annotating service properties.
- **Step 3:** Add semantic annotations to the marked service properties (Annotation Editor panel).

- **Step 4:** Locally save the service annotation as annotated HTML page or export it as extracted RDF (Semantic Description panel).

Consequently, SWEET can support the creation of semantic descriptions of RESTful services, appearing at the same time a flexible tool that facilitates the process of mashup creation.

### Summary and Related Approaches

Table 2 summarizes the main features of the three tools presented above. It also indicates their relation to the semantic technologies as well as the development approaches presented in previous sections.

In general terms, the *data triplification* technique seems to be common ground for every tool. The web-based application SWEET is by design a tool that its infrastructure relies exclusively on Semantic Web Services. As far as Jig4OWL is concerned, it does not appear to fall clearly within any of the approaches presented in Section *Methods and Techniques for Semantic Mashup Development*. However, the data triplification technique is an exception, since Jigs4OWL has access to data, represented by OWL DL ontologies, via SPARQL. Next, *Semantic Web Pipes* could initiate its operation by using the mashup language MashQL as a starting point. SWP could also utilize Semantic

Web Services, since the latter provide semantically annotated data in RDF format.

Other tools that can be loosely related to the scope of semantic mashups include also Sig.ma, Tabulator, Potluck and SPARQL Extensions. Sig.ma (Tummarello et al., 2010) is a simple and generic data aggregator that enables browsing and leveraging data from distributed and unrelated sources of the Web of data. Next, Tabulator (Berners-Lee et al., 2006), a semantic web browser (offered as an extension in Firefox) is met also in the literature about efforts for semantic mashup creation, since it is considered as a personal mashup tool and thus it cannot support development of large scale mashups. SPARQL extensions, proposed in (Nowack, 2008) in the context of the ARC2 RDF Store, reuse and enhance SPARQL's intuitive syntax in order to provide functionalities of data aggregation/update (SPARQL+), processing (SPARQLScripting) and output (SPARQL Result Templates). This effort leads to a set of appropriate tools for semantic mashup creation. Potluck (Huynh, Miller, & Karger, 2007) is a mashup creation tool available via Web and is appropriate for non-expert users without excluding programmers from its target group. This tool handles the retrieved data from Web sources in RDF format.

However, the above tools do not seem to make extensive use of semantic technologies to the extent to which their categorization as *semantic mashup* tools would be justified. Likewise, they do not follow a semantic-enabled development process in the sense given in the previous section.

## USE CASES

In this section, we present some real world use cases of semantic mashups, which cover different domains of interest in order to get a better idea of their utility. We have selected a couple of applications on e-tourism and e-commerce, although semantic mashups already exist for others significant real-life domains such as biomedicine (Bodenreider, Rutter, Skinner,

& Sheth, 2007), e-learning (Soylu, Wild, Mödritscher, & De Causmaecker, 2010) and more.

## Semantic Mashups for Tourism

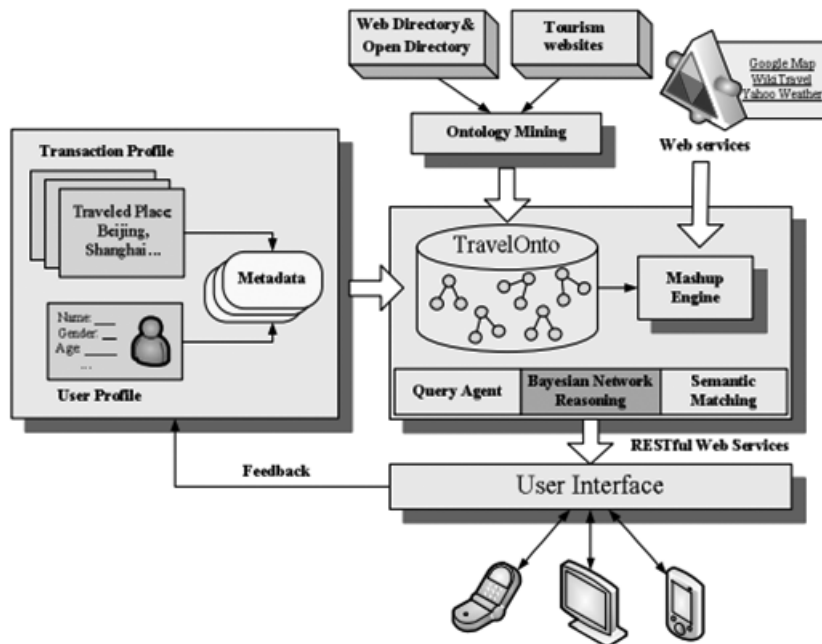
A main step in planning a trip includes the selection of the most attractive places to visit. To this end, a plethora of Web 2.0 tools have already been developed. Hence, travelers can easily access blogs, podcasts, wikis, map mashups and many other kinds of sources in order to share experience and research their travel adventure. All these systems take advantage of the notion of mashup, coming with the ability to aggregate data from different and various sources.

In this subsection, two well-known mashups for tourism, which take advantage of the great potentials of Semantic Web technologies, are described. The first one concerns an intelligent system based on ontologies. The second is focused on Linked Data consumption principles.

- **An Intelligent Semantic Mashup for Tourism:** This work proposes an intelligent recommendation system that illustrates how Semantic Web technologies can be exploited for data aggregation, rendering the creation of semantic mashups easier (Wang, Zeng, & Tang, 2010). This work has been applied in a real system, specialized in the tourist places of China. This system aims at the integration of heterogeneous online travel information (mashup development) and the recommendation of tourist attractions. This twofold purpose is accomplished by deploying two ontological models: the Travel Ontology and the User Ontology see Figure. 11.

The Travel Ontology constitutes a basic part in the proposed system's architecture and is the result of investigating many tourist websites and Web/Open directories in terms of structural information. This investigation retrieves information that actually points out the main terms used in the site map and the website menu. These terms are then utilized in constructing the basic concepts and their relationships in the resulting

Figure 11. Architecture of the China tourism system



ontological model. The emerging ontology is finally populated with data that are mashed up by spatial web services, such as Google Map, Yahoo Weather and WikiTravel.

Recommendation is achieved by developing a conceptual model about users' profiles. The User Ontology is intended to gather personal information, preferences, needs and interests about a user. Other ontology concepts, such as age, gender, profession and interests, are also taken into consideration in the construction of this model. In addition, several trip-specific concepts are added, so as to complement a user's profile or upgrade him to a traveling agent level. Finally, the system makes personalized recommendations by taking into account both the user's travel behavior as well as the travel behavior of other users and using a Bayesian network.

- **eZaragoza:** The project eZaragoza (Tejo-Alonso et al., 2010), deployed by Zaragoza's City Council, constitutes a tourist

promotional mashup that allows the access to tourist information from different data sources, both internal and external.

The eZaragoza mashup application consumes information about the monuments and historical buildings, restaurants, accommodation, shopping areas, cultural events and leisure activities from an internal source, called CRUZAR (Mínguez, Berrueta, & Polo, 2009). The latter is a different project, carried out also by Zaragoza's City Council, which makes all the aforementioned information available as Linked Data and exposes them via a SPARQL endpoint. This information is also enriched with spatial data and multimedia links.

On the other hand, the eZaragoza project retrieves additional data from non-Linked Data services, such as Flickr (for images' data), YouTube (for videos' data), Google Maps (for advanced spatial data), Yahoo (for weather forecasts) and Wikipedia (for document infor-

mation). Since these services are RESTful, the data retrieval is realized via proprietary APIs.

Moreover, several views of the offered data can be defined which are then handled as an autonomous user interface component, called gadget. For example, for an RDF resource, relevant to a monument entity, it is possible to define four discrete views, namely ‘multimedia view’, ‘factsheet view’, ‘map view’, and ‘document view’. These different views can be wired and arranged in a certain way, based on user’s personal preferences. So, each user has his own personalized interface to access the available data.

The overall function of gadgets is important in the case of eZaragoza project and is achieved via EzWeb (Lizcano, Soriano, Reyes, & Hierro, 2008). EzWeb is an open source mashup platform containing a generic catalogue of gadgets. In addition to the already developed ones, a user can create his own gadgets for the mashup under construction. Finally, in this mashup platform, any kind of gadget can be interconnected in order to exchange information.

## Books@HPClab Mashup

Books@HPClab mashup (Kalou, Pomonis, Koutsomitropoulos, & Papatheodorou, 2010) is an application that provides users with the ability to search and find metadata for books, which fit their personal preferences, from different data sources, like Amazon and Half eBay, see Figure. 12. All book information is made available as Linked Data in RDF format. Within this application’s framework users are considered autonomous entities whereas a separate user profile with their interests and preferences is maintained for each of them. So, different users can be treated differently, even though they are searching for the same key-phrase. The implementation of this application depends purely on ontology development, writing of rules (for the application’s personalization aspect) and data consumption from Web APIs of the aforementioned providers.

The application interacts with Amazon and Half eBay web services, by contacting

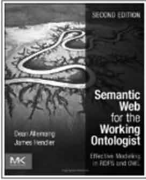
their respective Web APIs, so as to search and retrieve book metadata. The interaction with these APIs is based on the REST-protocol and the exchange of URL requests and XML files-responses. The application uses the RDF API for PHP (Cowles, 2004) in order to process aggregated data, to assign resolvable identifiers and ultimately to convert them to Linked Data in RDF format. The process of collecting data constitutes an application of the triplification method (sub-Section *Data Triplification*).

The set of collected data from bookstores and users’ profiles are captured and maintained in an ontology, named the BookShop Ontology. Book metadata offered by Amazon and Half eBay are mapped to this ontology. BookShop contains four main classes *Book*, *Author*, *Offer* and *User*, as well as a number of object and datatype properties. The first three classes represent concepts that are extracted from the structure and meaning of Amazon’s and eBay’s data. The class *User* is meant to express user profiles. The preferences of each user, such as preferable book condition, rating, publication year and maximum price are represented as datatype properties. Furthermore, object properties that link an instance of the class *Book* to an instance of the class *User* and inversely, are defined to express the user’s preference for a book depending on which preference fields of the user’s profile are covered. A set of DL-safe SWRL rules (Hitzler & Parsia, 2009), (“personalization rules”) is responsible for actually populating these properties.

These rules essentially distinguish those books, which satisfy user’s preferences from the entire set of books returned by Amazon and eBay. Initially, four rules were written to check the satisfiability of each preference criterion separately. For the case where two or three or four preference criteria are satisfied together, more rules have been devised, so as to check all possible combinations of satisfied criteria.

Books@HPClab mashup, in its latest version (Kalou, Koutsomitropoulos, & Solomou, 2013), is redesigned on top of the Drupal CMS by taking advantage of all the CMS features, such as front-end interface, user management,

Figure 12. Book information




**Semantic Web for the Working Ontologist, Second Edition: Effective Modeling in RDFS and OWL** ★★★★★

Morgan Kaufmann  
[Show/hide full description](#) or [See full details](#)

Available offers

ry\_books - \$36.58

Sold by ry\_books: Usually ships in 1-2 business days



**Programming the Semantic Web** ★★★★★

O'Reilly Media  
[Show/hide full description](#) or [See full details](#)

Available offers

Alin&Becky - \$17.84

Sold by Alin&Becky: Usually ships in 1-2 business days

dynamic content management, modular design and caching. The SWRL rules have been replaced by OWL 2 reasoning power to infer matching preferences.

## SUMMARY AND CONCLUSION

As much as fundamental the impact of advancing to Web 3.0 might be, it would be a quiet and natural transition; and this because it is not the technologies that are already available, which drive these developments, but a growing necessity for users and developers alike that inspires the cultivation of these technologies. The Semantic Web stack has already started to have a vertical influence in well-established Web 2.0 paradigms and mashup applications could have been no exception.

For once, rich descriptions for content and services, powered by expressive enough formalisms like RDF(S) and OWL, can aid the selection, filtering, recommendation and composition of data sources. In this spirit, a number of languages, design frameworks and systems have been proposed with the purpose to facilitate mashup design and development.

Likewise, several tools, GUIs and application frameworks have been implemented to provide out-of-the-box functionality and features for semantic mashup composition. These include for example approaches for semantic annotation of services, data source serialization based on RDF/OWL and components for semantic data consumption.

Even though with variable degrees of maturity and practical impact, all these approaches aim at taming content and services' semantics to make mashup development and use a better experience. The indicative applications presented as use-cases in this paper do just that and suggest the variety semantic mashups may have, ranging from intelligent web service orchestration to semantic-aided design to automatic data triplication and rules.

Semantic mashups may not exactly be the "killer-app" of the future Internet. However, they are definitely an interesting paradigm both of applying varied Semantic Web techniques as well as of the philosophy of next-generation web apps. We hope that this work would be a contribution towards the systematic examina-

tion, design, construction and assessment of semantic mashups.

## REFERENCES

- Adida, B., Birbeck, M., McCarron, S., & Pemberton, S. (2008). *RdFa in XHTML: Syntax and Processing, A collection of attributes and processing rules for extending XHTML to support RDF*. W3C Recommendation. (2008). Retrieved from <http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014/>
- Antoniou, G., Baldoni, M., Baroglio, C., Baumgartner, R., Bry, F., Henze, N., . . . Herzog, M. (2005). *Personalization for the Semantic Web II*. Technical Report IST506779/Hannover/A3-D4/D/PU/b1, Reasoning on the Web with Rules and Semantics, REWERSE
- Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., & Sheets, D. et al. (2006). *Tabulator: exploring and Analyzing linked data on the semantic web*. In *Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI06)*. Athens, Georgia.
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). *The Semantic Web*. In *Scientific American Magazine*. 33-43. Available online: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF2>
- Bianchini, D., De Antonellis, V., & Melchiori, M. (2010). A Recommendation System for Semantic Mashup Design. In *DEXA Workshops 2010*: 159-163 doi:10.1109/DEXA.2010.48
- Bin, L., Zhaohui, W., Yuan, N., Guotong, X., Chunying, Z., & Huajun, C. (2009). sMash: semantic-based mashup navigation for data API network. In *Proceedings of World Wide Web (WWW'09)*
- Bizer, C., Cyganiak, R., & Gauss, T. (2007). *The RDF Book Mashup: From Web APIs to a Web of Data*. In: *3rd Workshop on Scripting for the Semantic Web*
- Bizer, C., Heath, T., & Berners-Lee, T. (2009). *Linked Data – The story so far*. Special Issue on Linked Data [IJSWIS]. *International Journal on Semantic Web and Information Systems*, 5(3), 1–22. doi:10.4018/ijswis.2009081901
- Cowles, P. (2004). Exposing Web Applications Semantically Using RAP (RDF API for PHP). *PHP Architect*, 3(10), 34–39.
- Cyganiak, R., Wood, D., & Lanthaler, M. (Eds.). *RDF 1.1 Concepts and Abstract Syntax*. 9 January 2014. W3C Proposed Recommendation (work in progress). URL: <http://www.w3.org/TR/2014/PR-rdf11-concepts-20140109/>
- David, J., Euzenat, J., Scharffe, F., & Trojahn dos Santos, C. “The alignment API 4.0.” *Semantic web* 2, (2011) no. 1, pp 3-10
- Ding, L., Finin, T., Joshi, A., Pan, R., Scott Cost, R., Peng, Y., & Sachs, J. et al. (2004). *Swoogle: A Search and Metadata Engine for the Semantic Web*. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management* doi:10.1145/1031171.1031289
- Ehrig, M., & Staab, S. (2004). *QOM: Quick Ontology Mapping*. In *Proceedings of the 3rd International Semantic Web Conference (ISWC)*. 683-697
- Farrell, J., & Lausen, H. (2007). *Semantic Annotations for WSDL and XML Schema*. W3C Recommendation. Retrieved from <http://www.w3.org/TR/sawSDL/>
- Gomadam, K., Ranabahu, A., & Sheth, A. (2010). *SA-REST: Semantic Annotation of Web Resources*. W3C Recommendation. Retrieved from <http://www.w3.org/Submission/2010/SUBM-SA-REST-20100405/>
- Hitzler, P., & Parsia, B. (2009). Ontologies and Rules. In S. Staab & R. Studer (Eds.), *Handbook on Ontologies* (2nd ed., pp. 111–132). Springer. doi:10.1007/978-3-540-92673-3\_5
- Huajun, C., Bin, L., Yuan, N., Guotong, X., Chunying, Z., Jinhua, M., & Zhaohui, W. (2009). *Mashup by Surfing a Web of Data APIs*. In *Proceedings of the 35th International Conference on Very Large DataBases (VLDB 2009)*, Lyon, France
- Huynh, D., Miller, R., & Karger, D. (2007). *Potluck: Data mash-up tool for casual users*. In *The Semantic Web. In Lecture Notes in Computer Science* (Vol. 4825, pp. 239–252). Springer.
- Jarrar, M., & Dikaiakos, M. (2008). *MashQL: A Query-by-Diagram Topping SPARQL - Towards Semantic Data Mashups*. In *Proceedings of the 2nd International Workshop on Ontologies and Information Systems for the Semantic Web (ONISW)* part of the ACM CiKM conference. 89-96
- Jarrar, M., & Dikaiakos, M. (2012). A Query Formulation Language for the Data Web. *IEEE Transactions on Knowledge and Data Engineering*, 24(5), 783–798. doi:10.1109/TKDE.2011.41

- Jung, H., & Park, S. (2011). Mashup Creation Using a Mashup Rule Language. In *Journal of Information Science and Engineering*. Volume 27: 761-775
- Kalou, A. K., Koutsomitropoulos, D. A., & Solomou, G. D. (2013). *CMSs, Linked Data and Semantics: A Linked Data Mashup over Drupal for Personalized Search*. In *Proc. of the 7th Int. Conference on Metadata and Semantics Research (MTSR 2013)*. Springer. doi:10.1007/978-3-319-03437-9\_6
- Kalou, A. K., Pomonis, T., Koutsomitropoulos, D. A., & Papatheodorou, T. S. (2010). *Intelligent Book Mashup: Using Semantic Web Ontologies and Rules for User Personalisation*. In *Proceedings of the 4th IEEE Int. Conference on Semantic Computing (ICSC 2010) - Int. Workshop on Semantic Web and Reasoning for Cultural Heritage and Digital Libraries (SWARCH-DL 2010)*:536-541. doi:10.1109/ICSC.2010.78
- Khare, R., & Çelik, T. (2006). Microformats: a pragmatic path to the semantic web. In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin (Eds.), *WWW*: 865–866. ACM doi:10.1145/1135777.1135917
- Kitchen, D., Quark, A., Cook, W., & Misra, J. (2009). *The Orc Programming Language*. In *Proceedings of the Joint 11th IFIP WG 6.1 International Conference FMOODS '09 and 29th IFIP WG 6.1 International Conference FORTE '09 on Formal Techniques for Distributed Systems* doi:10.1007/978-3-642-02138-1\_1
- Kopecký, J., Vitvar, T., Fensel, D., & Gomadam, K. (2009). *hRESTS & MicroWSMO*. Technical report. Retrieved from <http://cms-wg.sti2.org/TR/d12/>
- Lathem, J., Gomadam, K., & Sheth, A. P. (2007). *SAREST and (S)mashups: Adding Semantics to RESTful Services*. In *Proceedings of IEEE Int'l Conf. Semantic Computing*. IEEE CS Press:469–476
- Lizcano, D., Soriano, J., Reyes, M., & Hierro, J. J. (2008). *EzWeb/FAST: Reporting on a Successful Mashup-based Solution for Developing and Deploying Composite Applications in the Upcoming "Ubiquitous SOA"*. In *Proceedings of the 2nd International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. UBICOMM 2008*, Valencia, Spain doi:10.1109/UBICOMM.2008.61
- Maleshkova, M. (2009). *Open Services on the Web*. Technical report kmi-09-5. Retrieved from <http://kmi.open.ac.uk/publications/pdf/report.pdf>
- Maleshkova, M., Pedrinaci, C., & Domingue, J. (2010). *Semantic Annotation of Web APIs with SWEET*. In *Proceedings of the 6th Workshop on Scripting and Development for the Semantic Web at Extended Semantic Web Conference*. Heraklion, Greece
- Manola, F., & Miller, E. (2004). *Resource Description Framework (RDF) Concepts and Abstract Syntax*. W3C Recommendation. Retrieved from <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
- Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., . . . Sycara, K. (2005). Bringing Semantics to Web Services: The OWL-S Approach. In: Cardoso, J., Sheth, A. (Eds.) *Semantic Web Services and Web Process Composition*, Lecture Notes in Computer Science, vol. 3387: 26-42. Springer Berlin / Heidelberg
- Matthew, R. (2010). *Data.dcs: from legacy to linked data*. In *Proceedings of Poster Track of the Extended Semantic Web Conference*
- Maximilien, E. M., Wilkinson, H., Desai, N., & Tai, S. (2007). *A Domain-Specific Language for Web APIs and Services Mashups*. In *Proceedings of the 5th International Conference on Service-Oriented Computing*. Volume 4749 of *Lecture Notes in Computer Science*, Springer-Verlag: 13-26 doi:10.1007/978-3-540-74974-5\_2
- Melchiori, M. (2011). *Hybrid techniques for web APIs recommendation*. In *EDBT/ICDT Workshop on Linked Web Data Management*: 17-23
- Mínguez, I., Berrueta, D., & Polo, L. (2009). CRUZAR: an application of semantic matchmaking to eTourism. In *Cases on Semantic Interoperability for Information Systems Integration: Practices and Applications* (pp. 255–271). IGI Global.
- Mitchell, I., & Wilson, M. (2012). *Linked Data: Connecting and exploiting big data*. White paper. Fujitsu UK
- Morbidoni, C., Le Phuoc, D., Polleres, A., Samwald, M., & Tummarello, G. (2008). *Previewing Semantic Web Pipes*. In *Proceedings of the 5th European Semantic Web Conference (ESWC 2008)*. Springer
- Nowack, B. (2008). *SPARQL+, SPARQLScript, SPARQL result templates – SPARQL extensions for the mashup developer*. In *Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC2008)*. Volume 401 de *CEUR Workshop Proceedings*



- Noy, N. F., & Musen, M. (2000). *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment*. In Proceedings of Seventeenth National Conference on Artificial Intelligence (AAAI), (TX US), Austin
- Noy, N. F., & Musen, M. (2003). The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping. *International Journal of Human-Computer Studies*, 59(6), 983–1024. doi:10.1016/j.ijhcs.2003.08.002
- Nuzzolese, A. G., Gangemi, A., Presutti, V., & Ciancarini, P. (2010). *Fine-tuning triplication with Semion*. In Presutti V, Svatek V and Share F (Eds) EKAW workshop on Knowledge Injection into and Extraction from Linked Data (KIELD2010) Le Phuoc, D., Polleres, A., Morbidoni, C., Hauswirth, M., & Tummarello, G. (2009). *Rapid semantic web mashup development through semantic web pipes*. In Proceedings of the 18th World Wide Web Conference (WWW2009), Madrid, Spain
- Prud'hommeaux, E., & Seaborne, A. (2008). *SPARQL Query Language for RDF*. W3C Recommendation. Retrieved from <http://www.w3.org/TR/rdf-sparql-query/>
- Sahoo, S. S., Bodenreider, O., Rutter, J. L., Skinner, K. J., & Sheth, A. P. (2008). An ontology-driven semantic mash-up of gene and biological pathway information: Application to the domain of nicotine dependence. *Journal of Biomedical Informatics*, 41(5), 752–765. doi:10.1016/j.jbi.2008.02.006 PMID:18395495
- Schwarzkopf, E., Bauer, M., & Dengler, D. (2003). *Towards intuitive interaction for end-user programming*. In Proceedings of the 8th international conference on Intelligent user interfaces (p./pp. 287--289), Miami, Florida, USA: ACM. ISBN: 1-58113-586-6 doi:10.1145/604045.604101
- Shvaiko, P., & Euzenat, J. (2013). Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1), 158–176. doi:10.1109/TKDE.2011.253
- Soylu, A., Wild, F., Mödritscher, F., & De Causmaecker, P. (2010). *Semantic Mash-Up Personal and Pervasive Learning Environments (SMupple)*. In Symposium of the WG HCI in Work and Learning Life and Leisure USAB: 501-504
- Sporny, M., Longley, D., Kellog, G., Lanthaler, M., & Lindstrom, N. (2014). *JSON-LD 1.0. A JSON-based Serialization for Linked Data*. W#C Available: <http://www.w3.org/TR/json-ld/>
- Tejo-Alonso, C., Fernández, S., Berrueta, D., Polo, L., Fernández, M. J., & Morlan, V. (2010). *eZaragoza, a tourist promotional mashup*. Submission to the AI Mashup Challenge 2010, co-located with the 7th Extended Semantic Web Conference (ESWC 2010)
- Tuchinda, R., Szekely, P., & Knoblock, C. A. (2008). Building Mashups by example. In Bradshaw JM, Lieberman H, and Staab S (ed) *Intelligent User Interfaces*: 139–148, ACM doi:10.1145/1378773.1378792
- Tummarello, G., Cygania, R., Catasta, M., Danielczyk, S., Delbru, R., & Decker, S. (2010). Sig.ma: Live Views on the Web of Data. In *Journal of Web Semantics: Science, Services, and Agents on the World Wide Web*, 8(4): 355-364
- Vitvar, T., Kopecký, J., Viskova, J., & Fensel, D. (2008). WSMO-Lite Annotations for Web Services. In S. Bechhofer, M. Hauswirth, J. Hoffmann, & M. Koubarakis (Eds.), *ESWC 2008. LNCS* (Vol. 5021, pp. 674–689). Heidelberg: Springer.
- Wang, W., Zeng, G., & Tang, D. (2010). Bayesian Intelligent Semantic Mashup for tourism. In *Journal of Concurrency Computation Practice and Experience*, Vol. 23, No. 8:850-862