STAR:chart – Preserving Data Semantics in Web-Based Applications

Irene Celino[1], Dario Cerizza[1], Francesco Corcoglioniti[1], Alberto Guarino[1], Andrea Turati[1], and Emanuele Della Valle[1,2]

[1] CEFRIEL, Politecnico of Milano, Via Fucini 2, 20133 Milano, Italy
{name.surname}@cefriel.it
[2] Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo, 32, 20133 Milano, Italy
emanuele.dellavalle@polimi.it

Abstract. Every time a Web-based interface is built upon pre-existing data-sources, the result comes often from a difficult and painful process of negotiation between the data manager – who knows the "semantics" of the data to be exposed, but cannot build the UI on his own – and the Web designer – who cares a lot about the presentation and the rendering, but often is not an expert about the content to be displayed.

Semantic Web technologies, with their ability to deal with data, their meaning and their representation, can be of great help in those scenarios. In this paper, we present our approach that makes joint use of Web usability, Web experience, Interaction Design and Semantic Web technologies to develop a Web-application framework called `STAR:chart`, which helps data managers in painlessly building Web-based interfaces upon their data.

We introduce the basic concepts, ontologies and software artifacts that make our approach concrete, together with some use case scenarios in which we successfully employed the `STAR:chart` framework.

1 Introduction

We definitely live in the era of the exponential grow of data scale [1]. Every company or organization has to deal with large datasets which are strictly related to its business and everyday operations. Moreover, the advent and spread of Web technologies provides a world-scale "repository" for the data storage, thus tempting or forcing organizations to expose their data on the Web.

In presence of pre-existing, large scale data sources, often stored in legacy systems, data managers must face the problem of "exposing", "externalizing" the data contained in those sources in order to let other people or organization access them and take advantage of their content. However, this operation is not always easy and painless: apart from the problems of security, privacy and confidentiality of data, data managers must deal with at least two different challenges. On the one hand, they must be able to *integrate* data coming from different sources, if possible creating cross-links between different pieces of information; on the other hand, they must assure that the new application giving access to those data is

W. Abramowicz (Ed.): BIS 2009, LNBIP 21, pp. 97–108, 2009.

usable from the final user point of view and that it *preserves data semantics*. In this paper, we will mainly refer to this second kind of problem.

Our claim is that Semantic Web technologies can facilitate this process of exposing data on the Web and to preserve data semantics in their presentation on the Web. Our long-time experience in the Semantic Web field made us build a lot of prototypes and demo applications to demonstrate the applicability and the usefulness of those technologies to solve real-world problems and challenges. In this paper we want to share our findings in using Semantic Web technologies to solve the aforementioned problem in both traditional and innovative ways: on the one hand we exploit the recognized ability of Semantic Web technologies to represent data and their meaning and to build human-readable and machine-readable applications, while on the other hand we demonstrate how ontologies and concepts/properties mappings can be successfully employed to formalize and to concretize the "semantics" of Web interaction and Web presentation, thus conducting to a new generation of Web applications that preserve the data meaning.

The remainder of the paper is structured as follows: Section 2 illustrates the scenario in which our approach is framed; Section 3 represents the core of the paper, with the explanation of our approach both at high level (Section 3.1) and in details, with the explanation of the defined ontologies (Section 3.2) and the characteristics of the STAR:chart framework (Section 3.3); Section 4 illustrates two real-world examples of employment of the STAR:chart framework in two different use cases; Section 5 presents some related works, while Section 6 concludes the paper and explains the further steps to improve our approach.

2 Problem Statement and Motivation

Currently the process of building a Web-based interface upon pre-existing data sources is a negotiation process between a data manager – who perfectly knows the meaning of the data, their structure and the way to access them in the repositories where they lay – and a Web developer or Web designer – who is not an expert in the data and cares only about the building of a user-friendly interface with its navigational paths.

Too often this negotiation ends with one of the parties getting closer to the understanding of the other party: either the data managers learn how to design and develop Web interfaces or the Web developers become an expert in the data field in order to understand what contents must be visualized in the final application. In other words, the difficulty lies in the background and understanding *gap* that separates the two parties.

We believe that a better scenario can be envisioned: the data manager should remain the "owner" of the knowledge required to understand the data meaning, but he should be also facilitated in creating a Web interface that satisfy his requirements. He should not be exposed to the problem of deciding about how the Web application is structured (e.g., in terms of areas, pages, sub-pages, menus, etc.), but he should be allowed to express what the purpose is of the application, what data should be presented, what concepts and resources are primary and what are accessory, etc.

Our approach relies on the definition of a set of primitives that are familiar to the data manager as well (and not only to the Web expert), that we call *fruition modalities*: if the data managers define the required fruition modalities that hold for his data, the process of building the Web interface that satisfies those needs becomes easier and (at least partially) can be made automatic, so that the required manual work of the Web developer to adjust the result to the data manager requirements is dramatically reduced.

Our idea is that the data managers express, in a declarative way, what "semantics" the Web application should convey: for example, he should define the central elements to be displayed; for each of them, he should list the identifying properties (used to name resources), the most relevant characteristics (used as "summary" or short description of a resource), the detailed attributes (displayed only if the reader wants to know more about a resource) and the related resources (to be linked together); moreover, for each core concept, the data manager should say what functionalities should be allowed over them (like searching, browsing or editing); and so on.

It is clear that this operation is quite easy for the data manager, since it does not require any particular skill in Web application design or development. Nonetheless, this "declaration of intents" is enough for an intelligent Web application framework to sketch a possible Web-based presentation over the data: each resource will be displayed with its identifying properties as "title" and its relevant characteristics as "summary"; if some detailed attributes are listed, a dedicated page should be created (otherwise only the summary will be displayed in some pages); if some related resources exist, they will be listed as links to other pages. Moreover, if a concept is defined as "searchable", a search box must be present to allow users to look for them; if a resource is marked as "editable", the user should be allowed to modify its content; and so on.

Our approach is based on the idea that the way the fruition modalities are translated into Web terms can be – at least partially – automated, by formalizing, through the use of ontologies, the *presentation and interaction semantics* on the one hand and the *Web design models* (like in model-driven Web design literature, as explained also in the following) on the other hand.

3 Our Proposal and Contribution for a Semantic User Interface

Since 2001, we have been investigating how Semantic Web technologies can improve the development and the user experience of Web applications. In particular, we conceived, designed and developed a framework to build knowledge-intensive information portals – called SOIP-F (Semantic Organization Information Portal Framework) – by employing ontologies and other Semantic Web technologies. The results of our research were published [2,3] within the scientific community and successfully deployed in several occasions (like the projects COCOON[1] and NeP4B[2]).

[1] COCOON (IST FP6-507126) integrated project.

[2] Italian FIRB project RBNE05XYPW NeP4B - Networked Peers for Business.

During the last few years, we refined and enriched the framework, both on the conceptual side and on the technical side. The current result is the STAR:chart framework, which, preserving the basic principles of SOIP-F and building on that experience, is an improved tool to help data managers to disclose their information sources by generating – in an easy and semi-automated way – a (Web) application to search and navigate across resources.

3.1 Overview of Our Approach

The innovation of STAR:chart consists in coupling *data semantics* (expressed through the use of domain-specific ontologies) with *presentation semantics* (expressed in our ontologies). We employ ontologies to formally represent the visualization and fruition of resources on the Web; this is enabled by the various studies in different fields like Web modeling, Web usability, user experience and interaction design. Section 5 lists some of the efforts from which we took inspiration when designing our system.

We developed two ontologies – as explained in the following section – to conceptualize the two points of view of the data manager and the Web developer: the STAR:dust ontology – the *presentation and interaction semantics* ontology – specifies the *fruition modalities* and the different roles the input data should play in the final application; the STAR:ship ontology – the *sitemap specification* ontology – specifies the primitives to model the structure of a Web application (taking inspiration from [4]) and represents the *application ontology* of the STAR:chart framework used at runtime to display the resources structured in Web pages.

Our claim is that, once the data manager expresses in a declarative way the desired role of his data in the final application, by the use of the STAR:dust ontology, the system can semi-automatically translate this declaration into a sitemap specification, expressed in STAR:ship terms. This is possible because each Web page can be divided into its basic elements (named travel objects elsewhere [5]), which are called *widgets* in our terminology; those widgets are on the one hand the realization of the fruition modalities (the various ways a user is able to interact with resources published on the Web like searching, browsing, tagging, etc.) and the building blocks of the sitemap on the other hand.

3.2 The STAR:dust and STAR:ship Ontologies

As introduced in the previous section and visually represented in the following Figure 1, the STAR:chart framework has two main underlying ontologies, namely STAR:dust and STAR:ship. This section is devoted to briefly explain the content and the purpose of those ontologies and the way they are employed in the framework functioning.

The **STAR:dust** ontology [6] is a conceptual model aimed at designing and specifying the navigation, that Web users undertake while surfing through resources. It provides a thorough conceptualization that can be used as *application ontology* (in a Model-driven Architecture approach) by the STAR:chart framework, which is a software tool that supports the navigation and the presentation of resources.

Therefore, the STAR:dust conceptual model specifies the *navigation and presentation semantics*. The resulting ontology, however, is not useful *per se*, but it is used to strongly decouple the editing of contents from their visualization. For example, once we have a domain ontology which describes the information contained in the data source, the data manager can "design" the information visualization by *mapping* between the domain ontology and the STAR:dust ontology. Finally, at runtime STAR:chart, taking as input both the domain knowledge and the mappings, makes lever on the STAR:dust ontology and produces a way to present and navigate across contents.

STAR:dust contains the primitive elements to define:

- the *fruition modalities*, i.e. the possible interactions of the final users with the presented information (search, browsing, detail view, tagging, editing, rating, etc.);
- the *presentation building blocks* as they are seen by a generic widget, i.e. the basic characteristics of data (relevant information, identifying features, graphic representation, spatial coordinates, filterable facets, groupable aspects, etc.);
- the *mapping approach*, i.e. how to express a mapping between a domain ontology and STAR:dust itself.

The data manager configuring the STAR:chart framework to present his data uses the STAR:dust ontology to create the *mapping definition*, the declarative artifact which states the "semantics" of data when they are displayed. In this mapping, the user specifies the role of concepts and properties of the domain ontology with regards to the presentation, by choosing a set of fruition modalities to be applied to information (e.g., the instances of the concept X should be browsable) and by mapping values to their visual appearance. An example of use of the STAR:dust ontology for the mapping definition[3] is offered in Listing 1 using N3 notation. In brief, this mapping says that, for the class `sf:Service` of the domain ontology, its property `sf:hasName` will play a double role in the Web-based presentation of all the instances of the aforementioned class: on the one hand, it will be used as title property (`dc:title`) every time a page will be devoted to that resource description; on the other hand, it is an important characteristic (`sd:RelevantProperty`) that should be presented as "identifying" attribute every time a resource of that type is included in a page.

```
:sampleMapping  a  sd:PresentationMapping ;
        sd:onClass             sf:Service ;
        sd:mappingSource       sf:hasName ;
        sd:mappingDestination  dc:title ;
        sd:mappingDestination  sd:RelevantProperty .
```

Listing 1. Example of mapping with STAR:dust

[3] Prefix `sd` indicates the STAR:dust ontology, prefix `dc` the Dublin Core metadata vocabulary, while prefix `sf` refers to Service-Finder ontology, cf. Section 4.1.

The **STAR:ship** ontology, on the other side, is the conceptual model that defines the structure of a Web application in terms of pages, widgets and related artifacts. Modeling a high-level description of a Web site under various dimensions (i.e., structure, composition, navigation, layout and personalization) is not a new idea. The STAR:ship ontology, as a consequence, builds on the long-term research and know-how in Web engineering; in particular, our conceptual model takes inspiration from the well-known and industrially-exploited WebML language [4], from which it borrows the concepts of area, page, unit and link.

At runtime, apart from the mapping definition, the framework needs a model of the sitemap, i.e. an abstraction of the Web site which contains all the elements that make the framework manage efficiently users' requests. The sitemap is at a lower level of abstraction with regards to the data mapping, it is the artifact needed at runtime. Therefore, when configuring the STAR:chart framework with a specific data source, a *sitemap specification* is needed. This sitemap is expressed with regards to the concepts and properties of the STAR:ship ontology and should be derived by the mapping definition itself as well as from the description of widgets and fruition modalities of the STAR:dust ontology. An example[4] is given in Listing 2; this fragment of sitemap specification says that the instances of the `sf:Service` class must be listed in the `searchResultsPage`; to this end, a `ss:ListWidget` is used and a specific template will be exploited for the visualization. A `ss:ListWidget` is configured to display a list of resources and for each of them its `sd:RelevantPropertys`. This means that, at run time, whenever the framework will be asked to display this `serviceListWidget`, it will query the data source to retrieve the `sd:RelevantPropertys` of each listed instance of the class `sf:Service`, as defined in the mapping definition (so that, for example, property `sf:hasName` will be retrieved, cf. Listing 1).

```
:serviceListWidget  a  ss:ListWidget ;
      ss:onClass       sf:Service ;
      ss:hasName       "service list" ;
      ss:hasParent     :searchResultsPage ;
      ss:hasTemplate   "service_list.ftl" .
```

Listing 2. Example of sitemap specification with STAR:ship

In the current implementation of the STAR:chart framework, however, this automatic generation of the sitemap specification is not yet available and therefore this artifact must be provided by the framework user together with the mapping definition. It is in our research agenda the further investigation on this topic: we will enhance (if needed) the two main ontologies and we will find an automated (or semi-automatic) method to generate the sitemap specification from the existing artifacts, in order to reduce the work of customization of the framework, hence easing its adoption and use.

[4] Prefix `ss` in this case indicates the STAR:ship ontology.

3.3 The STAR:chart Framework

A high-level abstract representation of STAR:chart is offered in Figure 1. The framework is able to perform the following functionalities:

1. accessing the data through an abstraction of *data source*
2. preserving data semantics when "translating" resources expressed in the domain ontology to resources expressed in the presentation ontology
3. generating the appropriate Web pages to navigate across resources

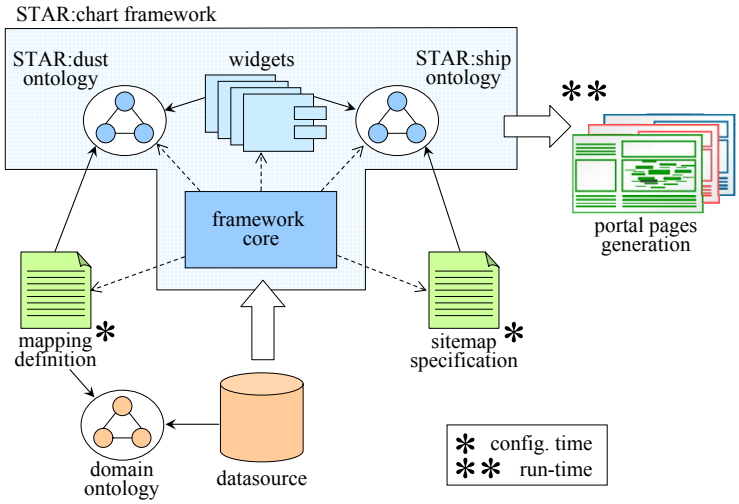

Fig. 1. High-level representation of the STAR:chart framework

Item 1 in the previous list is enabled at configuration time by the connection to the data source and by the "injection" of the domain ontology in the framework. This operation lets STAR:chart access the data abstracting from the specific data repository: the framework uses the data source abstraction to send requests as query-by-criteria[5] and to get the results in RDF format, leaving to the specific data source implementation the (eventual) task of translating queries and converting results. To ease its adoption, the framework provides reference implementations of this data-source abstraction for common sources like RDF repositories.

Item 2 is enabled, at configuration time, by the definition of mappings between the domain ontology and the presentation ontology (STAR:dust). This declarative artifact lets the data manager specify what the purpose is when displaying the information: he can express that some characteristics of data (such

[5] The previous versions of the framework directly submitted SPARQL queries, since they assumed the interaction with a plain RDF store; we extended the system by creating the *Data Source* interface in order to make the framework more generic and able to deal with different kinds of back-end systems. See Section 4.1 for an example.

as properties applied to a concept in the domain ontology) have a specific role when being visualized (e.g., a title, an image, a spatial information) or can be used for performing specific actions (e.g., filtering, searching, comparing, editing). This artifact – the *mapping definition* – is almost the only configuration task requested to the data manager when using the framework. On the basis of this mapping, the most suitable widgets to present the mapped resources can be selected and configured to be used at runtime.

The last item of the previous list, at configuration time, needs the specification of the structure of the sitemap expressed with reference to the STAR:ship ontology (see Section 3.2). In the current stage of the STAR:chart implementation, this *sitemap specification* must be manually provided; our current research, however, focuses on the possibility to automate, at the cost of a loss in expressive power, this specification, by extracting the needed information from the mapping definition (which contains the preferences of the data manager in terms of fruition modalities) and from the STAR:dust and STAR:ship ontologies themselves; the latter ones can formalize some standard compositions of widgets into pages to reflect users' needs and to respect presentation/visualization best practices in terms of usability.

Finally, the generation of Web pages is enabled at runtime by the framework that provides (1) the selection of the suitable widgets on the basis of the sitemap specification, (2) the translation of each widget's information needs in a query over the data source, (3) the access to the data source and (4) the aggregation of the widgets and their rendering in Web pages.

4 Case Studies

In this section, we provide some insights in the use we made of the STAR:chart framework in developing two different applications. The former is related to the design and development of the Web interface of the Service-Finder EU research project, while the latter regards an interface for a logistic scenario as in one of the use cases of the NeP4B project.

4.1 Service-Finder

Service-Finder is an EU co-funded research project, which aims at building a Web 2.0 environment for Web service discovery. Service-Finder will increase the efficiency of service provision and subsequently enable service usage with the ambitious but achievable goal of making the service-oriented-architecture scale to millions of services on the Web. For more information, please visit Service-Finder project homepage: `http://www.service-finder.eu/`

In the context of Service-Finder we employed the STAR:chart framework to design and build the Web-based User Interface, from now on called Service-Finder Portal (SFP). In the following, we briefly explain the configuration steps of the STAR:chart framework we performed to build the Service-Finder Portal on top of it.

The instantiation of the STAR:chart framework in Service-Finder requires:

- the *configuration of the data source*: in the Service-Finder project, the role of data source is played by the Conceptual Indexer and Matcher, which is the component used to store and index all the information about the services. The Conceptual Indexer and Matcher component is based on OntoBroker [7] and Lucene [8]; to this end, an integration API was defined: it expects a query in the STAR:chart framework format and returns results in RDF format;
- the *selection of the fruition modalities* and related characteristics to better present the main objects of Service-Finder (mainly services and providers, with the correlated entities); to this end, we created the mapping definition which contains the various mappings between the Service-Finder ontologies[6] and the STAR:dust ontology;
- the *sitemap specification*, in terms of pages and widgets[7], in order to enable the framework to generate the needed UIs; we designed this sitemap manually, but in the future we will be able to synthetize it starting from the mapping definition and the framework ontologies.

The result can be publicly accessed at `http://demo.service-finder.eu`.

4.2 NeP4B

NeP4B – Networked Peers for e-Business – is an Italian Government co-funded research project aimed at building an infrastructure to help "business peers" to improve and facilitate their business in a coopetitive (cooperative and competitive) scenario. For more information, please visit the NeP4B project homepage at `http://www.dbgroup.unimo.it/nep4b/`

In this context, we employed the STAR:chart framework to build the User Interface of a "peer" in a logistic use case. In this case, the STAR:chart framework acted as a *Semantic Navigation Engine*, to allow browsing and searching through truckload demand and offer. In the logistic use case, different logistic operators expose their data about their vehicles and their planned travels with correspondent goods loads; since those operators are small-medium enterprises, a single "peer" is built by the integration of the data coming from the different actors. The STAR:chart framework is used to build the Web application that gives access to the demand and offer, thus leading to a more effective truckload exchange.

This was made possible by mapping to the STAR:dust ontology the data structures and by defining the search and browse fruition modalities required over those data. The result is a multilingual Web portal on which it is possible to search and browse through load requests and load offers, preserving and

[6] The Service-Finder ontologies are published on the Web, respecting the Linked Data publishing blueprints [9,10], at `http://www.service-finder.eu/ontologies/Service-Ontology` and `http://www.service-finder.eu/ontologies/Service-Categories` respectively.

[7] In Service-Finder, we extended the framework to add Web 2.0-style widgets for collaboration, which previously were not part of the framework.

exploiting the different dimensions of data: the results can be browsed through traditional lists, filtered by faceted browsing and visualized geographically on a map or temporally on a timeline.

For confidentiality reasons of the real logistic operators involved, the public portal available at `http://seip.cefriel.it/truckload-exchange` is only a demonstrative application with faked data.

5 Related Works

As briefly outlined throughout the paper, our work takes inspiration from several approaches in different communities, like Web usability, user experience, interaction design, Web 2.0, etc. In the Semantic Web field, a number of proposals try to address similar problems providing partial or extensive solutions. Without aspiring to being exhaustive, in this section, we provide some pointers to the relevant literature.

The centrality of data meaning preservation is stated also by [11], which argues that, since RDF enables data from multiple sources to be easily integrated to form a single view on a particular topic, traditional Web pages can be considered dead. Semantic Web applications giving access to RDF data (such as Semantic Web browsers) should stop working at the level of pages and start focusing, instead, on the level of "things". This includes supporting different kinds of interactions depending on the object the user is interested in. Available actions and data sources could even change in accordance with the specific task or context, requiring sophisticated models to enable selective integration of heterogeneous sources.

[12] presents Fresnel, an RDF vocabulary which aims at solving the aforementioned issues. Fresnel is based on two concepts: *lenses* and *formats*. Lenses specify which properties of RDF resources are shown and how they are ordered, whereas formats describe how to present the information selected by lenses, optionally adding extra static content. Fresnel's goal is to supply a browser-independent RDF vocabulary which enables users to define how Semantic Web content is presented and is general enough to be applicable across applications.

Designing navigation and presentation tools under the form of "widgets" is the purpose also of Mondeca ITM (Intelligent Topic Manager) with its Semantic Portal and Semantic Widgets [13]. They offer a portal solution based on a widget library: display elements, to be added in a portal, which offer advanced functionalities (natural language requests, search results display and navigation tools, faceted navigation, results maps, subjects relationships mapping). The difference with our STAR:chart approach lies in the abstraction and decoupling layer we inserted through the mapping definition (see Section 3.2) that clearly separates the data layer from the presentation layer.

A similar approach is the one proposed in [14] (and following publications), called OntoViews; it is built upon Apache Cocoon framework and produces valid RDF/XML (instead of plain XML), which is displayed by the use of stylesheets and XSLT. The selection of "views" is however done case-by-case on the basis

of the specific data. Another solution, this time based on the Alfresco CMS, is the one developed by [15] for the Cantabria Cultural Heritage Semantic Portal.

All those approaches and solutions can be considered further evolutions of the early works on Semantic Portal such as the well-known SEAL [16], on which is based the website of Institute AIFB. We also owe much to their work; nonetheless, our claim is that we do not only provide a way to build a single Semantic Portal, but a framework to be used to build domain specific Web applications that, as a plus, can be exploited also by data manager without a strong background in Web technologies, helping them preserving their data semantics.

6 Conclusions and Future Work

In this paper, we presented our innovative approach to build Web-based user interfaces and applications to access pre-existing data sources. We explained the problems data managers face today when they decide to expose their data on the Web and the challenges in this context.

We propose a new approach that makes use of ontologies to improve the final application design and development and we introduced the framework we built to demonstrate the feasibility and the goodness of our approach. We also illustrated two real-world applications that successfully make use of the STAR:chart framework.

The main innovation in our work is the employment of formalized "semantics" in the form of ontologies to express not only the meaning of data but also the navigation, presentation and interaction models and their significance in Web design and development. We believe that our approach is not only scientifically and technically sound, but it also has a pragmatic advantage that results in a business value: the agreement between a data manager and a Web designer can be reached easily and painlessly by the use of common abstractions (like the fruition modalities we introduced) that decreases the gap between the two actors points of view. Therefore, the building of a Web interface is less expensive in terms of time and effort.

Our future works, as outlined in the course of the paper, will be devoted to continue our investigation about the automation of the process. The sitemap specification explained in Section 3.2 must be performed manually today, but we believe that it can be largely made automatic by extending the core part of the STAR:chart framework. Moreover, in order to make our solution general and flexible and to fulfill the needs of the largest possible user base, we are continuously adding more predefined widgets to our implementation. The interested reader can keep an eye on our work by following the updates and the new finding at `http://swa.cefriel.it/STAR`

Acknowledgments

This research has been partially supported by the *Service-Finder* EU co-funded project (FP7-IST-215876) and by the *NeP4B* Italian Government co-funded FIRB project (MIUR-2005-RBNE05XYPW).

References

1. VVAA: Big Data. Specials – Nature News 455(7209), 1–50 (2008)
2. Della Valle, E., Brioschi, M.: Toward a framework for semantic organizational information portal. In: Bussler, C.J., Davies, J., Fensel, D., Studer, R. (eds.) ESWS 2004. LNCS, vol. 3053, pp. 402–416. Springer, Heidelberg (2004)
3. Celino, I., Della Valle, E.: Multiple Vehicles for a Semantic Navigation Across Hyper-environments. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 423–438. Springer, Heidelberg (2005)
4. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. In: Proceedings of the Ninth International World Wide Web Conference, Amsterdam, Netherlands. Elsevier, Amsterdam (2000)
5. Yesilada, Y., Stevens, R., Goble, C.A.: A foundation for tool based mobility support for visually impaired web users. In: WWW, pp. 422–430 (2003)
6. Celino, I., Della Valle, E., Corcoglioniti, F.: The STAR:dust conceptual model. W3C SWD SKOS Use Cases and Requirements Material (2006), `http://www.w3.org/2006/07/SWD/wiki/EucStarDustDetailed`
7. Ontoprise: Ontobroker 5.0. user manual. Technical report (2007), `http://www.ontoprise.de/`
8. Gospodnetic, O., Hatcher, E.: Lucene in action. Manning Publications (2004)
9. Berrueta, D., Phipps, J.: Best Practice Recipes for Publishing RDF Vocabularies – W3C Working Draft (2008), `http://www.w3.org/TR/swbp-vocab-pub/`
10. Bizer, C., Cyganiak, R., Heath, T.: How to Publish Linked Data on the Web (2007), `http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/`
11. Heath, T.: How will we interact with the web of data? IEEE Internet Computing 12(5), 88–91 (2008)
12. Pietriga, E., Bizer, C., Karger, D.R., Lee, R.: Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 158–171. Springer, Heidelberg (2006)
13. Mondeca: Semantic Portal - Semantic Widgets (2008), `http://www.mondeca.com/index.php/en/intelligent_topic_manager/applications/semantic_portal_semantic_widgets`
14. Makelä, E., Hyvönen, E., Saarela, S., Viljanen, K.: OntoView – A Tool for Creating Semantic Web Portals. In: Proceedings of the 3rd International Semantic Web Conference (ISWC 2004), Hiroshima, Japan (2004)
15. Hernandez Carrascal, F., Rodrigo, L., Contreras, J., Carbone, F.: Cantabria Cultural Heritage Semantic Portal (2006), `http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-295/paper02.pdf`
16. Maedche, A., Staab, S., Studer, R., Sure, Y., Volz, R.: SEAL - Tying Up Information Integration and Web Site Management by Ontologies. IEEE Data Engineering Bulletin 25(1), 10–17 (2002)