

# DISTRIBUTED METAINFORMATION SEARCHING: AN APPROACH TO INFORMATION RETRIEVAL IN THE AGE OF THE SEMANTIC WEB

David F. Barrero  
University of Alcalá  
email: dfbarrero@wanadoo.es

Diego R. López  
RedIRIS  
Edif. CICA - Avda. Reina Mercedes s/n, 41014 Seville, Spain  
email: diego.lopez@rediris.es

Óscar García Población  
Computer Engineering Department, University of Alcalá  
Escuela Politécnica Superior, Despacho E-313  
Ctra. Madrid-Barcelona km. 31.6, Alcalá de Henares, Spain  
email: oscar@aut.uah.es

## Abstract

Semantic representation of information has some evident advantages from an integration point of view, but there are many situations in which this approach is not directly applicable, as information is stored in several formats without (implicit or explicit) knowledge of its semantics. We present a distributed middleware able to unify the access to any arbitrary information source by doing a semantic translation of its contents.

## 1 Introduction

In the context of knowledge management, the location and access of information is a key element for success in an organisation. The best results of this process are achieved when the organisation does not manage information, but knowledge. From this point of view, information is confined to a single resource whereas knowledge may be distributed in many resources, so information in a resource should contain some reference to its context. This is what, in computational terms, we may call semantics.

The reality in most organisations is that information is not stored using the best way to integrate information systems. This information is usually spread, stored in incompatible formats, and often strongly coupled to its presentation. It goes from single files to information managers such as data bases or any other backend system, each one with its own interface;

therefore, integrating them forces the development of middleware layers, often complex and expensive.

Information is usually stored without any indication about its context and purposes, and this is the origin of many search problems that would not exist if we had some notion about that context, as it implies limiting the search to a set of discrete resources. Efficient management of information requires viewing it as a whole, being able to manage its context: it is necessary to deal with semantics to cross the line between managing information and managing knowledge.

This paper presents a search solution aimed to achieve a double objective: integrating the location of information resources and expressing them in semantic terms by means of a compatible format with web technologies in general and the emerging web services in particular.

## **2 Technological framework**

The WWW was conceived in the late eighties with the aim of sharing information among different research groups, and the amazing growth of the web along the nineties is a proof of its success. However, this growth has created a set of new problems. The volume of the stored information is too high to allow an efficient management with traditional means. In this line, search systems have evolved from brute force to more intelligent systems that deal with knowledge [19], but there is a quite difficult natural border to overcome in knowledge retrieval, due to the nature of the technologies used in the Web.

The Web keeps a huge amount of information in HTML pages, but this information has no context and thus it cannot be considered as knowledge. Hiperlinks cannot be considered as a context because they do not modify the nature of data meaning: in practical terms they only expand the web page size.

Another problem derives from the fact that the Web is based on the concept of information, while it is used quite often for service access, like e-commerce web sites, and this usage of the Web is constantly growing. The interface to provide services in the web is typically a web form that the user has to fill out, i.e., two similar services may have quite different interfaces. Again, we have to face up the problem of integrating different services with different interfaces, leading to a problem conceptually similar to the one we had described for information integration.

All these problems have their origin in that the Web was designed originally to be used by human beings. When the Web had a relative small size traditional means to represent information based on raw HTML was an useful approach, but today, with the huge size that the Web has reached, it has a new set of problems that must be overcome and users request new features that must be provided.

Automation of information and service processing in the Web is a highly desirable objective, and may produce a new generation of Web applications and new usage experiences. Two main technologies have been envisaged to face this problem.

### **2.1 Semantic Web**

The basis of the above described problems is the lack of a separation between the information itself and the representation of that information and, on the other hand, the fact that the format of the information does not provide any clue to computers about its meaning, so, automation of information processing is a great problem in the traditional Web.

Tim Berners-Lee proposed what he called the semantic web [17], a new paradigm in which information in the web is described using not only representation but also semantical terms.

This is an evolutionary process during which the Web of information will be transformed into the Web of knowledge, complementing the existing web pages rather than replacing them.

Semantic representation of information provides the possibility of a revolution in the way users employ the Web, thanks to the automation of information processing. Instead of manually browsing the Web, we may use a personal software agent and instruct it to do some task [7]. For example, to give us the best way to travel between two cities at the lowest price suitable in our agenda, a software agent might use and compare the semantic description of the time-table of different transport means and its price.

The two core technologies of the semantic web receive the name of RDF (*Resource Description Framework*) [4] and OWL (*Web Ontology Language*) [10].

RDF defines a syntax to describe semantic networks where we have several linked statements with a subject, a predicate and an object, that is what the literature about RDF calls a triple. Different triples can be linked creating a semantic network. It is used to represent knowledge in a formal way that a computer can “understand”. A quite informal graphical example of the RDF model can be found in figure 1, it has two triples stating “Homer Simpson lives in Springfield” and “Homer Simpson likes beer”.

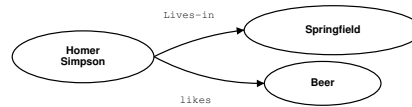


Figure 1: A simple RDF example

Since RDF only defines a syntax, it does not state anything about the vocabulary that is used, its semantic and the relationship between the elements of the vocabulary, i.e., the ontology.

The mean to describe ontologies in the Web is OWL. RDF combined with OWL defines a model of knowledge that is understandable by computers and that may establish the future’s new relationship between users and the Web.

## 2.2 Web Services

According to the World Wide Web consortium a web service is “a software system designed to support interoperable machine-to-machine interaction over a network” [6]. This machine-to-machine interaction in the Web contrasts with the traditional human-to-machine interaction used in the Web.

In the context of web services is the machine, and not the human, the entity accessing services, using a set of protocols based on XML to perform RPC calls, typically SOAP<sup>1</sup> [5]. The great promises of this approach are the integration of services in heterogeneous environments and the automation of service access. The interface to a web service may be formally described in a machine understandable format, like WSDL (*Web Services Description Language*) [13].

With web services, for example, developing a complex web site to help the planification of journeys may be quite easy. We could use a self-made service of path selection, check the traffic status with the appropriate public authority, and finally integrate the weather forecast service given by a television channel’s web site.

A real example of web services application may be found in Google. To access Google’s search facilities, an user typically makes use of a browser and performs the query on its web

<sup>1</sup>SOAP V1.1 recommendation stands SOAP acronym as Simple Object Access Protocol, but the more modern SOAP v1.2 recommendation does not stand an official acronym for SOAP.

site. Google offers also access to its search engine through a web service [1] in which any application can obtain the same results of a manual query in Google's web site.

The idea of web service is being complemented by a more general idea, the *service-oriented architecture* (SOA). This architecture proposes a framework in which a web service may be discovered and invoked automatically using its interface description. The most usual protocols to perform those tasks are *Universal Description Discovery Integration* (UDDI) [18] for discovery, WSDL for interface description and SOAP for invocation.

Coming back to the example of a journey planification web site, this does not need to use a fix set of services: the application may discover a weather forecast service from a public UDDI directory, find out its interface with WSDL, generate automatically a client for the web service, and invoke it with SOAP.

Web services have the great advantage of being loosely coupled services, in contrast with traditional distributed object systems like CORBA, DCOM or RMI, that are tightly coupled. A web service may change its interface, adding for example a new parameter, and this change does not involve any modification in the clients.

### 2.3 Joining up Web Services and the Semantic Web

Web services and the semantic web have been independently evolving. Web services are an effort of the industry to modularize the services in Internet meanwhile the semantic web has been developed in an academic environment. As both technologies mature, an increasing work in order to combine them is being done.

Research in this area has focused on the semantic discovery of web services [16, 14] as the way to integrate web services in the semantic web. Instead of using a directory based system like UDDI, it uses the semantic web technologies to discover web services. Using this approach it is possible to integrate processes into the semantic web, in other words, web services become part of semantic information within the semantic web.

Directly related to the semantic discovery of web services is the semantic composition of web services, in which semantic description of web services is used to compose several simple web services and as result we obtain a complex one.

Our proposal is not the integration of the information about the service but the integration of the information that the service uses. From this point of view, there is not many work done [11]. It basically entails using web services with RDF parameters.

## 3 Objectives

The diversity of information support systems, the lack of knowledge about their semantics and the coupling of container and contents makes information sharing and knowledge management a complex task, complexity that grows exponentially with the amount of information. Thus, it is useful to find mechanisms able to provide a homogeneous interface to search and locate heterogeneous information stored in diverse support systems.

Most of the information support systems offer search interfaces that facilitate information location in such system, but they are generally dependent on the stored information contents and therefore are incompatible among them. A way to solve this problem is to develop a search abstraction layer in which search operations can be integrated, providing an homogeneous search interface: in other words, federating search services.

Joining up the semantic web with web services is an excellent framework to provide a solution to the problem of information retrieval, which makes it possible to perform semantic search in an architecture neutral system. Many different search services and information

support systems can be accessed in a unified transparent way, building a quite promising technology that could achieve general use in a short period of time.

## 4 System Description

### 4.1 Overview

Our proposal to comply to the objectives described in the previous section is a distributed search system called Searchy, able to integrate searches in an arbitrary number of search systems. It has been designed to be extensible with a highly modular architecture, as a result, support for new search systems can be easily added.

Searchy has been designed to make it usable by data base administrators, web masters and other IT staff whose main tasks are not related to information system integration, so ease of use has been one of the key design goals of the software. Searchy has a simple configuration interface and it is intended to be an autonomous software component that does not require an application container -very common in the web services world- like Tomcat. Being autonomous provides an easier configuration and the administrator can focus on the application matters, instead of wasting time with esoteric container properties. Simplicity is also a key feature to avoid security flaws.

Searchy is, in a typical installation, composed of several independent agents working cooperatively by searching information and providing its results to other agents or applications. An agent in this context is a traditional server, that uses SOAP, and whose main task is to query one or more information support systems or other agents.

Functionally, we can divide Searchy into three components:

- The Searchy core. It is the common part for all the agents. It manages incoming requests and network related issues, reads the configuration and sets up the data providers. In general, the core performs all the tasks not specific for a information provider.
- The provider. It manages the access to a single data support system and builds metadata from it. In general, Searchy may be seen as a container of providers, since they provide the functionality and modularity that the application requires.
- The data support system. It is the information that is accessed. It may be a data base manager, a directory service, a HTTP server, local files or even another Searchy agent.

We can consider Searchy as a special case of three tier applications, in which the provider acts as the middle tier, although offering some extra functionality.

As it has been described, Searchy is not aimed to be used by end users, it is just a middleware, and it is supposed that other programs will use it; for example, the client may be a custom servlet to integrate Searchy in a web site. However, Searchy provides some classes to develop Java clients as well as Python clients with no need to deal with any SOAP details. A generic client is provided with Searchy, quite useful to test Searchy's configuration.

Each agent publishes its services as SOAP web service, so they can be used from any architecture and any language that supports SOAP, like C, C++, Java, Python, PHP, Perl and others. Web services may be used in any environment to the point that one may use Searchy services from any language able to invoke SOAP calls, and therefore it is not limited to have a web interface. It can be implemented as a servlet, a PHP page or a heavy graphical application. Several prototypes of Searchy clients are currently available: thin Java clients, servlets, JSPs and a GUI-enhanced application made in Python.

The administration of Searchy is done with a configuration file, in the same way as a typical server, as long as it is a self-contained application. This configuration file uses a XML format and is easy to understand and use for any administrator.

## 4.2 Data Formats

The generality of the information to be accessed and the range of its possible formats is quite high, thus we cannot directly manage it. Instead of working with the “raw” information, we work with a concept that fits better to this problem: metainformation expressed in semantic terms. All the system works with a model of metainformation that abstracts local information formats, representing some properties about the resources it locates, i.e., Searchy is a metainformation search engine.

An important mission of the agent is to translate metainformation queries into some query understood by the local data support system and, in the same way, to obtain metainformation from the information given by the data support system. From this point of view, an agent may be considered as a gateway with some extra features.

Metainformation alone is not knowledge, it is only information about information. To transform metainformation into knowledge we need to provide a context, that is, a semantics understandable by computers. In this point is where semantic web technologies come into action.

The metainformation format uses the core technology of the semantic web, RDF. Our first aim is to locate resources by means of integrating search systems, so we can be quite restrictive in the vocabulary we use, because we only describe resources. Just for this purpose the Dublin Core Metadata Initiative created a set of fifteen terms, called Dublin Core [8], whose objective is to describe anything that has an identity. Dublin Core only defines the semantic of a set of terms, so that it may be expressed with different syntaxes, like HTML[9, 12] or the one that interests us, RDF [3].

The combination of RDF with Dublin Core provides a general model valid to describe any resource from many different points of view. It conforms the core of Searchy data and, since it is coded in XML, it is extremely portable. It is also the core idea supporting the transformation of data into metadata, and of metadata into knowledge. This data model is the basis of the whole system. Even queries to the system are coded in RDF with a Dublin Core vocabulary.

## 4.3 Providers

Getting metainformation from the information stored in the data support is a task for the information provider and it has a strong dependence on the properties of the data support system. There are some support systems, for example some text formats, that have some metainformation integrated in the document, and we may use them. But the general case is when we have to obtain the metainformation from the stored information, directly mapping information into metainformation.

The access to the specific information provider must be, clearly, hard coded in the provider, and a different provider must be developed in order to access a different data support. One of Searchy’s main design objectives was the simplification of the development of providers, the system provides a set of facilities and services, so the developer can focus in his business.

Extracting metainformation from the information provider is, by far, the most difficult task for the system’s administrator. In general, the information is obtained from any field or property of the resource because usually information formats do not keep metainformation integrated in. It is the administrator’s decision to map information into metainformation. Searchy helps in this task by giving a string substitution mechanism for which each metainformation

field (in other word, each Dublin Core term) may be composed by none, one or more information fields. This approach has shown to be valid in all the data support systems considered for the moment. However, the modular design of the application eases the implementation of different translation mechanisms.

Agents have been designed to be highly extensible, therefore, adding new information supports might be quite easy, and the flexibility of the system facilitates the implementation of a wide variety of providers, there are very few limitations.

At the moment three types of information providers have been successfully implemented: SQL, LDAP and Google. Searchy agents allow to find information in an homogeneous and transparent form addressing data bases with SQL, LDAP directory services or even doing Internet wide searches using the Google API.

There is a fourth sort of implemented information provider, the RDF provider. It has an architectural functionality that allows recursive structure to the layout of the application. The RDF provider is an example of the flexibility of this approach, all inter-agent interaction is done this way.

#### 4.4 System layout examples

The underlying philosophy of Searchy makes it very flexible and may be used in a wide range of situations and with a wide range of aims. The concept of provider and its implementation lets us easily add new information supports. The possibility of communication between different agents is the key to distribute the application, and it is a powerful feature for the administrator. Let us call system layout to the structure composed by the logical interconnections between the different agents of a Searchy system.

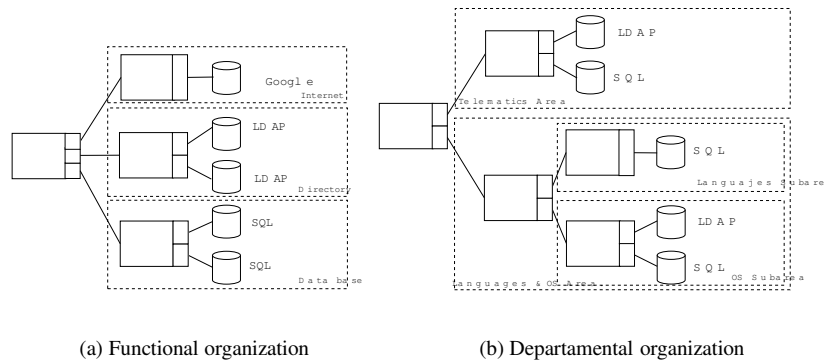


Figure 2: Searchy layout

A realistic layout may be found in figure 2-a. It depicts a functional layout in which each agent is responsible for dealing with one type of information support systems. In this layout, the management of SQL and LDAP agents could be a responsibility of the database administrator and the directory administrator, who are the people with the best understanding of each information system. A Google agent has also been included in the example, not only due to its real applicability but also to show the power of this approach of federating heterogeneous sources for information retrieval.

The functional approach shown above may not be suitable in great organisations, where a departmental distribution may be a better solution. Figure 2-b shows a typical structure of an

university department. It has two areas and one of the areas has two subareas, being each one responsible to manage an agent that will publish the data they provide. This recursive structure may be as complex as needed, with the only limitation that the overall performance is given by the longest path in the layout.

#### 4.5 Conclusions and future work

Searchy is a scalable, modular and highly distributed search system that provides search service federation as well as a framework for distributed information retrieval on which it is possible to develop new search interfaces, data mining applications, or any other information retrieval system. The flexibility of Searchy makes it suitable to solve a wide range of problems in many different situations and it is distributed under a free license.

At the moment, Searchy is in a beta state of development, it is fully functional and quite robust. Three providers have been developed: SQL, LDAP and Google API; there is a fourth provider, the RDF provider, in practise, a Searchy client within an agent used to access to other agents services. As long as Searchy has been implemented with JDBC and JNDI, it is able to access a wide range of data bases and directory services. Searchy even may access Internet wide resources using the Google search engine.

Using Dublin Core as a fixed vocabulary is a reasonable approach to describe resources, but in order to improve flexibility and its range of applications, full support for any RDF vocabulary should be implemented.

In its current state, one of Searchy's strongest limitations is its query format, based on raw RDF, and without an abstraction of the query model to have a better performance when using data providers query syntaxes. To solve this problem in the next future Searchy would have support for RDQL (*RDF Data Query Language*)[15], a query language similar to SQL. With RDQL Searchy will be able to manage complex data searches, with no limitation in the vocabulary it uses, although at the price of more complex provider implementations.

Searchy has a modular design, so it can be easily expanded by developping new providers, and from this perspective Searchy offers a wide range of applications. In an immediate future, a Harvest [2] provider will be developed. Harvest is a distributed search system used in Internet and intranets, with support to collect information from many formats. Another interesting provider to be developed is a general web service provider, that discovers and invokes search web services dynamically. The discovery may be done with a well known technology like UDDI or through a semantic discovery system or any other protocol.

### 5 Acknowledgements

We would like to thank Javier Masa Marín, researcher of RedIRIS and developer of Harvest, for his support all along this project, as well as José Manuel Macias, also member of RedIRIS for his help. We also thank Juan Ignacio Criado, PhD student, for his contributions to this paper.

### References

- [1] *Google APIs*. <http://www.google.com/apis/>.
- [2] *Harvest project home page*. <http://harvest.sourceforge.net/>.
- [3] Eric Miller Dave Beckett and Dan Brickley. Expressing simple dublin core in rdf/xml, July 2002. <http://dublincore.org/documents/dcmes-xml/>.

- [4] Dave Beckett (editor). *RDF/XML Syntax Specification (Revised)*. W3C Recommendation, February 2004. <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>.
- [5] Nilo Mitra (Editor). Soap version 1.2 part 0: Primer. Technical report, W3C Recommendation, June 2003. <http://www.w3.org/TR/soap12-part0/>.
- [6] Hugo Haas and Allen Brown (editors). Web services glossary. Technical report, W3C Working Group, February 2004. <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>.
- [7] James Hendler. Agents and the semantic web. *IEEE Intelligent Systems Journal*, 16(2):30–37, March-April 2001.
- [8] Dublin Core Metadata Initiative. *DCMI Metadata Terms*, November 2003. <http://dublincore.org/documents/dcmi-terms/>.
- [9] J. Kunze. Encoding dublin core metadata in html. RFC2731, December 1999. <http://www.ietf.org/rfc/rfc2731.txt>.
- [10] Deborah L. McGuinness and Frank van Harmelen. *OWL Web Ontology Language Overview*. W3C Recommendation, February 2004. <http://www.w3.org/TR/2004/REC-owl-features-20040210/>.
- [11] Uche Ogbuji. *Using RDF with SOAP*, February 2002. <http://www-106.ibm.com/developerworks/webservices/library/ws-soappdf/>.
- [12] Andy Power. *Expressing Dublin Core in HTML/XHTML meta and link elements*, November 2003. <http://dublincore.org/documents/dcq-html/>.
- [13] Jean-Jacques Moreau et alia Roberto Chinnici, Martin Gudgin. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. W3C Working Draft, November 2003. <http://www.w3.org/TR/2003/WD-wsdl20-20031110>.
- [14] T. Son S. McIlraith and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, March/April.
- [15] Andy Seaborne. *RDQL: A Query Language for RDF*, January 2004. <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>.
- [16] Siegfried Handschuh et alia Tanja Sollazzo. Semantic web service architecture - evolving web service standards toward the semantic web. In Susan M. Haller and Gene Simmons, editors, *FLAIRS Conference*, pages 425–429, Pensacola, May 2002. AAAI Press.
- [17] James Hendler Tim Berners-Lee and Ora Lassila. The semantic web. *Scientific American*, 284(5):28–31, May 2001.
- [18] Luc Clement Tom Bellwood and Claus von Riegen (editors). *UDDI Version 3.0.1*. UDDI Spec Technical Committee Specification.
- [19] Osmar R. Zaiane. From resource discovery to knowledge discovery on the internet. Technical Report TR 1998-13, Simon Fraser University, August 1998.

## **Vitae**

### **David F. Barrero**

David is a student of Telecommunications Engineering at the University of Alcalá and student of Computer Science at the UNED. He is author of several publications and has been consultant in the area of public management. His research interests are focused in the Semantic Web and e-administration.

### **Diego R. López**

Diego R. Lopez is the responsible for the Middleware & Applications Area of RedIRIS, the Spanish NREN. He graduated in Physics at the University of Granada in 1985, and earned his PhD degree at the University of Seville in 2001. In 1985, he joined the Conformance Testing Division of Telefonica I+D, working in several projects related to e-mail and directory services. From 1992 to 2000 was member of the technical staff of CICA, acting as the responsible for network services. In January 2000 he joined RedIRIS. His current work is focused on Internet middleware, specially on semantic Web interfaces and security-enhanced services.

### **Óscar García Población**

Óscar García Población was born in Avila, Spain in 1973. He has a three-year degree in Computer Engineering from the University of Alcalá and a six-year degree in Computer Science from the Universidad Politécnica of Madrid. During the last five years he has been working on industrial computing and traffic control. Since 1994 he has been a lecturer on Operating Systems in the Computer Engineering Department of the University of Alcalá. His interests include real-time system development, industrial control and traffic management.