The Semantic Web Framework: a component-based framework for the development of Semantic Web applications

Raúl García-Castro, Asunción Gómez-Pérez and Óscar Muñoz-García
Ontology Engineering Group. Departamento de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid, Spain
{rgarcia,asun,omunoz}@fi.upm.es

Abstract

Semantic Web technology development and reuse is difficult for people outside the research community. This paper presents the Semantic Web Framework, a structure in which Semantic Web applications can be organised and developed. This component-based framework contains the definition of the semantic-related software components that can be used in the development of Semantic Web applications, the dependencies that exist between these components, and the existing implementations of components that can be used in this framework.

1. Introduction

Semantic Web technology is being used beyond the borders of the research world and is reaching all kinds of users ranging from companies to individuals. These users, after discovering the benefits of the Semantic Web technology, want to make a step further and switch from technology consumers to technology producers.

However, when users try to develop Semantic Web applications they have to face several obstacles. They do not know neither the types of technologies now existing or the functionalities that these provide, nor do they know the dependencies between the different technologies. It is not easy to know how to use the existing Semantic Web technology and how to reuse or include this technology into users own applications. They do not know if these technologies can interoperate either between themselves or with their own technologies and, if so, how this interoperability can be achieved. And they cannot accurately make decisions, such as cost or resource estimations, when including semantic capabilities into their applications or when building Semantic Web applications from scratch.

An universal agreement on how to develop a Semantic Web application is impossible, but it is possible to facilitate the understanding and development of semantic web applications exploiting software reuse techniques.

The Semantic Web Framework is intended to help Semantic Web application developers build Semantic Web applications and solve all these problems. It is a reference framework that: (1) describes the existing types of Semantic Web technologies, their functionalities, and the dependencies between these technologies; (2) facilitates technology reuse by providing specifications and guidelines; (3) shows how to achieve interoperability with Semantic Web technology; and (4) helps make decisions when developing Semantic Web applications.

The Semantic Web Framework classifies the different Semantic Web technologies according to their functionalities and represents them as independent components. It provides a description of the functionalities that these components offer and also provides the dependencies between these components.

This paper is structured as follows: Section 2 describes the features of Semantic Web applications as well as existing proposals of architectures for this kind of applications. Section 3 describes the Semantic Web Framework and the components involved in it. Finally, Section 4 draws the conclusions of this work and proposes future lines of work.

2. Semantic Web Applications

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [1]. In this context, where the web is a network of application-usable information, we can define a Semantic Web Application as a software application that uses and produces information for the Semantic Web.

Semantic Web Applications have been characterized by different authors [6] and by events such as the Semantic Web Challenge[1] with the following features: (a) data has

[1]`http://challenge.semanticweb.org/`

semantics and is represented using formal descriptions, (b) semantic data is reused, manipulated and processed, (c) data sources are heterogeneous and are owned or controlled by different organisations, (d) applications assume an open world (i.e. the information is never complete), (e) multiple natural languages are supported, and (f) RDF(S) and OWL, the open standards recommended by the W3C, are used.

In addition, Motta and colleagues define the features for a next generation of Semantic Web Applications [6], which are: (a) semantic data can be defined in terms of many different ontologies, (b) Semantic Web Applications must scale in terms of the amount of data used and in terms of distributed components working together, and (c) Semantic Web Applications ought to embed Web 2.0 features.

In the Semantic Web, reuse appears not only at the data level, as shown above, but also at the application level, as nowadays there exist many open software from a wide range of sources that can be reused when building Semantic Web Applications. In the application level reuse follows three different approaches: a distributed services approach, by integrating web service technology in their architectures; a shared memory approach, by composing components that use a shared space of common memory to communicate, as in the case of reusing libraries inside an application; and a mixed approach, by combining the two approaches explained before.

2.1. Semantic Web Application Architectures

Mika et al. sketch a generic architecture of ontology-based applications based in a call-and-return style and structured in hierarchical layers [5]. The layers involved are from bottom to top: ontology, middleware and application. The ontology layer contains the components concerned with the creation and maintenance of the model of the application, the middleware layer supplies common ontology-related services, and the application layer builds on the ontology and related services to provide some kind of ontology functionality to an end user.

Thanh et al. [9] present a service oriented architecture also structured in hierarchical layers: the data layer hosts any kind of data sources including others different from ontological sources; the logic layer includes application-specific services that are implemented for a particular use case and operate on specific object models; the presentation layer hosts presentation components that the user interacts with. They also classify the components inside the logic layer in ontology services, ontology engineering services and ontology usage services.

The Semantic Web Technology is composed by heterogeneous systems. Therefore, the framework described in this paper is an open system and it is not divided in layers. Several disadvantages of layered approaches are the difficulty in structuring some systems in a layered fashion; performance considerations when high level functions require close coupling to low level implementations; and the difficulty to find the right level of abstraction, especially if existing systems cross several layers [7].

The two architectures presented before identify some example components for illustrating their approaches. In the Semantic Web Framework, we have tried to exhaustively identify the existing semantic components of Semantic Web applications. The 33 components identified in the Semantic Web Framework cover the 16 and 21 components identified in the previous approaches, respectively.

3. The Semantic Web Framework

In this paper, the Semantic Web Framework is defined as a structure in which Semantic Web applications can be organised and developed. The Semantic Web Framework is guided by some general design principles. These principles state that the Semantic Web Framework should be:

- *Developer-oriented.* To consider different audiences such as developers with low expertise with Semantic Web technologies or ontology practitioners.

- *Easy to understand.* To facilitate the understanding and use of the Semantic Web Framework, its components have been organised in dimensions according to the major properties of the problem space that have significant variation over Semantic Web technology.

- *Inexpensive to adopt.* To develop a Semantic Web application or to upgrade an existing application with semantic capabilities should be easy and thus, the impact on legacy systems minimised.

- *Semantics focused.* To describe only the components that provide semantic functionalities and functionalities to manage semantics. Other components that deal with communication, distribution, etc. are not taken into account to ease the integration of the components of the Semantic Web Framework in other software architectures.

- *Component based.* To define some specifications of these components that allow for different implementations of them, providing each of these components a basic functionality.

- *Evolving.* To extend easily the Semantic Web Framework by inserting new components or modifying existing ones as the Semantic Web, and also its technology, is continuously evolving.

The Semantic Web Framework is defined as a component-based framework because Semantic Web applications possess similar characteristics to those of component-based systems: interoperability, distribution, heterogeneity, extensibility independence, and dynamism. Furthermore, component-based frameworks provide the features that facilitate software reuse [4]: *abstraction*, to reduce and factor out details; *selection*, to help developers locate, compare and select reusable software artifacts; *specialisation*, to allow specialising generic artifacts; and *integration*, to combine a collection of artifacts.

A software architecture is defined as the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [3]. According to this definition of software architecture, to define the architecture of the Semantic Web Framework we need to identify its components, the interaction between them, the patterns that describe their composition, and the restrictions when applying those patterns.

In this paper, we focus on the identification of the components of the Semantic Web Framework; on their classification, as stated below; and on the main interfaces of these components with other components and with the environment. In a future work, we will define a concrete specification of the interfaces and the different patterns that can be used in Semantic Web applications.

3.1. Definition and Classification of Components

We follow the definition of component given by Szyperski [8] since a Semantic Web Framework component is as an autonomous and modular unit with well defined interfaces that describes a service and performs a specific functionality. Such components can be used either independently or together to develop Semantic Web applications. Components in this sense can be divided into four types: services, program libraries, applications, and protocols.

Components are usually defined by specifying some *general information* about them, such as a natural language description; their *interfaces*, including the functionalities that the component implements and those that it uses; and their *contracts*, which are specifications that are added to the interface and establish use and implementation conditions [8].

In this version of the Semantic Web Framework, we do not describe the component contracts, which will be defined in future work, and we explicitly divide the interfaces in the functionalities that a component implements and those that it uses. Therefore, each component is defined by its *name*, a high-level *description*, an enumeration of the *functionalities that the component provides* specifying for each functionality the type or types of interface that it provides (user interface, programming interface, service interface, etc.), and an enumeration of the *functionalities required by the component* for working correctly and provided by others.

To classify the components of the Semantic Web Framework, we have considered the dimensions of an architecture as the major properties of the problem space that have significant variation over the systems of concern to the architecture, i.e., the groups of components that provide some specific support to the architecture. These dimensions are subjective: in this paper we have classified the different components according to the main functionalities that they provide. Furthermore, these dimensions are not exhaustive.

Figure 1 presents the components of the Semantic Web Framework that have been identified from software currently available or under construction. This enumeration of components is neither exhaustive nor complete, and is open to improvements and extensions. Current components were identified by several Knowledge Web partners with expertise in each of the dimensions.

In Figure 1, each dimension of the architecture is represented as a column and the order of the components or of the dimensions in the figure does not imply any precedence or relation between them.

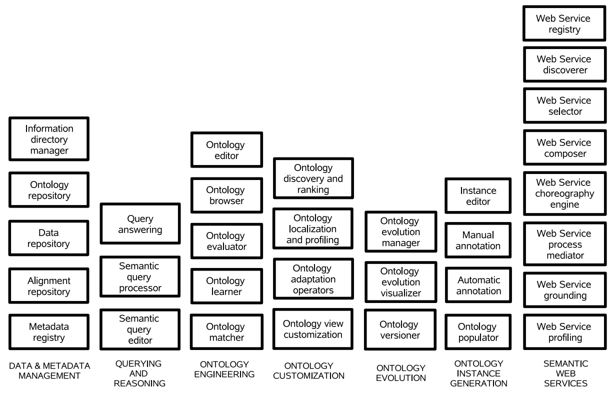

Figure 1. Components of the Semantic Web Framework

The dependencies of each of these components with other components of the framework were identified. Figure 2 shows the basic dependencies of the components in the *Ontology engineering* dimension.

Next, a description of the dimensions of the Semantic Web Framework and of the components included inside each dimension is given. The full description of the Semantic Web Framework components can be found in [2].

3.1.1. Data and metadata management

This dimension includes components that manage knowledge and data sources.

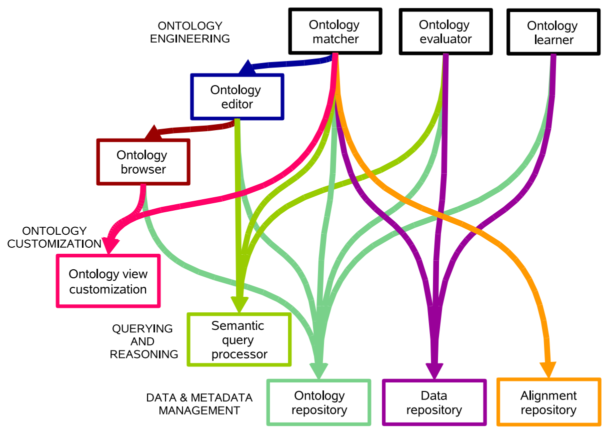

Figure 2. Dependencies of the components in the *Ontology engineering* dimension

The *Information directory manager* handles query distribution, manages a content provider directory, identifies information providers from a query, and handles the storage and access to distributed ontologies and data.

The *Ontology repository* locally stores and accesses ontologies and ontology instances.

The *Data repository* locally stores and accesses data and ontology annotated data.

The *Alignment repository* locally stores and accesses alignments.

The *Metadata registry* locally stores and accesses metadata information.

3.1.2. Querying and reasoning

This dimension includes components that generate and process queries.

The *Query answering* takes care of the logical processing of a query by providing reasoning functionalities to search results from a knowledge base.

The *Semantic query processor* takes care of the physical processing of a query, by providing functionalities to manage query answering over ontologies in distributed sources.

The *Semantic query editor* takes care of the user interface for editing queries.

3.1.3. Ontology engineering

This dimension includes components that provide functionalities to develop and manage ontologies.

The *Ontology editor* allow to create and modify ontologies, ontology elements, and ontology documentation. These functionalities include a single element edition or a more advanced edition such as ontology pruning, extension or specialization.

The *Ontology browser* allows to visually browse an ontology.

The *Ontology evaluator* evaluates ontologies, either their formal model or their content, in the different phases of their life cycle.

The *Ontology learner* acquires knowledge and generate ontologies of a given domain through some kind of (semi)-automatic process.

The *Ontology matcher* matches two ontologies and outputs some alignments. We can distinguish two types of such systems: those that generate matchings and those that use matchings for other tasks (e.g., merging, mediating, etc.).

3.1.4. Ontology customisation

This dimension includes components that customise and tailor ontologies.

The *Ontology localization and profiling* adapts an ontology according to some context or some user profile.

The *Ontology discovery and ranking* findins appropriate views, versions or sub-sets of ontologies, and then ranks them according to some criterion.

The *Ontology adaptation operators* applies appropriate operators to an ontology, the result of which is an ontology customized according to some criterion.

The *Ontology view customisation* enables the user to change or amend a view on a particular ontology to fit a particular purpose.

3.1.5. Ontology evolution

This dimension includes components that manage the ontology evolution.

The *Ontology versioner* maintains, stores and manages different versions of an ontology.

The *Ontology evolution visualizer* visualizes different versions of an ontology.

The *Ontology evolution manager* manages the adaptation of an ontology regarding changes that may arise and possibly visualises the versions within a broader context of complex ontology evolution and development platform.

3.1.6. Ontology instance generation

This dimension includes components that generate ontology instances.

The *Instance editor* allows to manually create and modify instances of concepts and of relations between them in existing ontologies.

The *Manual annotation* is in charge of manual and semi-automatic annotation of digital content documents (e.g. web pages) with concepts in the ontology. This annotation process may be assisted or guided by a machine (semi-automatic annotation).

The *Automatic annotation* automatically annotates digital content (e.g. web pages) with concepts in the ontology.

The *Ontology populator* automatically generates new instances in a given ontology from a data source.

3.1.7. Semantic web services

This dimension includes components that manage semantic web services.

The *Web service discoverer* publishes and searchs service registries, controls access to registries, and distributes and delegates requests to other registries.

The *Web service selector* checks whether the services can actually fulfil the user's concrete goal and under what conditions.

The *Web service composer* is in charge of the automatic composition of the web services in order to provide new value-added web services.

The *Web service choreography engine* uses the choreography descriptions of both the service requester and provider to drive the conversation between them.

The *Web service process mediator* reconciles the public process heterogeneity that can appear during the invocation of web services.

The *Web service grounding* is responsible for the communication between web services.

The *Web service profiling* creates web service profiles based on their execution history.

The *Web service registry* registers semantic web services.

4. Conclusion

The Semantic Web Framework is intended to help developers build Semantic Web applications and to diminish the cost of this development. This paper is a first step for providing foundation for the large-scale development of Semantic Web applications, by presenting a first definition of the Semantic Web Framework, describing the existing types of Semantic Web technology, their functionalities, and the dependencies between these technologies.

Although the Semantic Web Framework is useful as a reference and helps reusing existing technology, Semantic Web application developers will still have to develop their applications and their functionalities.

Immediate uses of the Semantic Web Framework include the identification of the components needed for a Semantic Web application in the design phase or the identification of existing implementations of components to be reused.

A first validation of the Semantic Web Framework has been performed by defining the eight use cases in Knowledge Web using the components of the Semantic Web Framework. These definitions can be found in [2].

One extension of the Semantic Web Framework is to include a new dimension for social components. Work in this direction is being performed in the Avanza project PLATA.

Another line of work will be to develop specifications of the components, of their life cycle, and of their interfaces; as well as guidelines for implementing these components and for reusing them when developing Semantic Web applications.

Acknowledgements

Thanks to the collaborators in the definition of the Semantic Web Framework components: S. Costache, S. Dasipoulou, Y. Ding, M. Dzbor, J. Euzenat, M. Kaczmarek, F. Lécué, D. Maynard, V. Novacek, R. Palma, R. Piskac, D. Zyskowski, L. Nixon, A. Léger, T. Vitvar, M. Zaremba, and J. Hartmann.

This work is partially supported by a FPI grant from the Spanish Ministry of Education (BES-2005-8024), by the IST project Knowledge Web (FP6-507482), by the CICYT project Infraestructura tecnológica de servicios semánticos para la web semántica (TIN2004-02660), and by the Avanza project PLATA (FIT-350503-2007-6).

References

[1] T. Berners-Lee, J. Handler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[2] R. García-Castro, O. Muñoz-García, M. Suárez-Figueroa, A. Gómez-Pérez, S. Costache, D. Maynard, S. Dasiopoulou, R. Palma, V. Novacek, F. Lécué, Y. Ding, M. Kaczmarek, R. Piskac, D. Zyskowski, J. Euzenat, M. Dzbor, L. Nixon, A. Lger, T. Vitvar, M. Zaremba, and J. Hartmann. D1.2.5 Architecture of the Semantic Web Framework v2. Technical report, Knowledge Web, December 2007.

[3] IEEE. *IEEE Std 1471-2000. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. IEEE, September 2000.

[4] C. W. Krueger. Software Reuse. *ACM Comput. Surveys*, 24(2):131–183, June 1992.

[5] P. Mika and H. Akkermans. D1.2 Analysis of the State-of-the-Art in Ontology-based Knowledge Management. Technical report, SWAP Project, February 2003.

[6] E. Motta and M. Sabou. Next Generation Semantic Web Applications. In *Proc. of the 1st Asian Semantic Web Conference (ASWC)*, September 2006.

[7] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, April 1996.

[8] C. Szyperski. *Component Software, Beyond Object Oriented Programming*. Addison-Wesley, 1998.

[9] T. Tran, P. Haase, H. Lewen, Muñoz-García, Ó., A. Gómez-Pérez, and R. Studer. Lifecycle-Support in Architectures for Ontology-Based Information Systems. In *Proceedings of the 6th International Semantic Web Conference*, pages 508–522, November 2007.