Towards a Component-Based Framework for Developing Semantic Web Applications

Raúl García-Castro[1], Asunción Gómez-Pérez[1], Óscar Muñoz-García[1], and Lyndon J.B. Nixon[2]

[1] Ontology Engineering Group, Departamento de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid, Spain
{rgarcia,asun,omunoz}@fi.upm.es
[2]AG Netzbasierte Informationssysteme, Freie Universität Berlin, Berlin, Germany
nixon@inf.fu-berlin.de

Abstract. For those outside the research community, to develop Semantic Web applications entails real difficulty. This difficulty is due in part to the lack of usable approaches for planning Semantic Web solutions, even though Semantic Web tools have already reached industrial maturity. We propose here the Semantic Web Framework, a component-based framework for analysing rapidly the required components, the dependencies between them, and selecting existing solutions. This approach has been tested with a number of industrial partners, which justifies the effort made in this direction.

1 Introduction

Semantic Web technologies are slowly but surely moving out of the borders of the research community and reaching all types of business users, ranging from large multinational companies to individuals. These users, when convinced of the benefits that the Semantic Web technology provides to their problems and processes, may want to switch from being technology consumers to technology producers, by building their own Semantic Web-based solutions on top of existing tools and methodologies. However, when non-expert users try to plan and develop Semantic Web solutions they currently face several obstacles:

- They do not know the types of technologies now existing nor the functionalities that these provide, nor do they know what are the dependencies between the different technologies.
- They do not know how to use the Semantic Web technology, so they cannot reuse or include this technology into their own applications.
- They do not know whether these technologies can interoperate either between themselves or with their own technologies and, if so, how this interoperability can be achieved.
- 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.

J. Domingue and C. Anutariya (Eds.): ASWC 2008, LNCS 5367, pp. 197–211, 2008.

Although reaching a universal agreement on how to develop a Semantic Web application is almost impossible, facilitating the understanding and development of Semantic Web applications by giving design guidelines through software patterns and exploiting software reuse techniques is really feasible. Nowadays, to construct applications from a collection of reusable components and frameworks is a popular approach to software development. Components provide a number of benefits because they simplify application development and maintenance, and thus, they allow systems to be more adaptive and to respond rapidly to changing requirements [1].

The Semantic Web Framework is intended to help Semantic Web application developers design and build Semantic Web applications. This framework can be a first step to solve the above problems, though later on it should be extended with interface descriptions, benchmarking, interoperability tests and cost models. The framework is a reference framework that currently provides descriptions of the existing types of Semantic Web technologies and their functionalities, and of the dependencies between these technologies.

Our approach involves classifying the different Semantic Web technologies according to their functionalities and representing them as independent components grouped under a smaller set of component groups. For each component, we give a description of the functionalities that the component provides and then we identify the dependencies between the different components. The level of the descriptions is understandable enough to non-experts; additionally, with our industry partners we have validated through use case analysis the accessibility of the framework to non-experts, enabling them to identify rapidly the required components with their planned Semantic Web application, thus ensuring a viable final concept through taking component dependencies into account.

With the appropriate extensions to the framework, we expect to facilitate the use and reuse of this technology and to avoid inconsistencies when developing Semantic Web applications by providing further specifications and guidelines for components.

This paper is structured as follows: Section 2 presents a brief explanation of component-based software development, software architectures and frameworks. Section 3 describes the commonalities of Semantic Web applications and the related work that supports application development in this context. Section 4 focuses on the Semantic Web Framework, the components involved in it and in the dependencies between such components. Section 5 shows how the Semantic Web Framework is used to support real industrial use cases and to determine their component needs and dependencies. Finally, Section 6 draws the conclusions of this work and proposes future lines of research.

2 Background

2.1 Component-Based Software Engineering

Reuse-based software engineering is becoming the main development approach for business and commercial systems. One of this reuse-based approaches is

Component-Based Software Engineering (CBSE), which is the process of defining, implementing and composing loosely coupled independent components into systems [2]. In CBSE, application developers reuse components already developed and tested to build their applications in a robust and rapid way, only knowing the component interface or contract and not knowing the details of the component implementation or the way the component was conceived to be used.

CBSE relies on *independent components* that are completely specified by their interfaces, *component standards* that facilitate the integration of components, *middleware* that provides software support for component integration and *a development process* that is geared to CBSE. According to this, the Semantic Web Framework provides the skeleton for a specification of the independent components needed.

A software component is a software composition unit that specifies a set of interfaces and a set of requirements; and that can be composed with other components independently in time and space [3].

Component-based systems have the following characteristics:

- *Interoperability.* Components cooperate despite differences in language, interface, and execution platform.
- *Distribution.* Components can be hosted in different machines in a network.
- *Heterogeneity.* Components can be executed in different platforms or operating systems and written in different languages by different developers.
- *Extensibility independence.* The applications are modifiable and extensible adding new components.
- *Dynamism.* Applications can evolve by component extension, extinction, substitution, or by reconfiguring the relationships between components.

The Semantic Web Framework has been defined as a component-based framework because Semantic Web applications possess similar characteristics to component-based systems above presented. 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 particularize generic artifacts; and *integration*, to combine a collection of artifacts.

2.2 Software Architectures and Frameworks

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 [5].

The objectives of software architectures are to understand and improve complex application structures; to reuse application structures so as to solve similar problems; to plan the application evolution; to analyse the application correction and the compliance degree with respect to the initial requirements; and to allow the study of some domain specific parts.

Software architectures are described by a) the *components* that realise the computational and data storage aspects; b) the *interaction* between components

during the execution; c) the *patterns* that describe the component composition; and d) the *restrictions* imposed when applying those patterns.

Frameworks are a kind of domain-specific software architecture [6], which define the architectural style relating the components inside a system. Furthermore, they define a set of components and their interfaces in an abstract way, establishing the interaction rules and mechanisms between them.

Depending on the framework applicability, frameworks can be classified into horizontal and vertical frameworks [3]. *Horizontal frameworks* are valid for every application domain relative to a concrete aspect of the system (e.g., communication infrastructures, user interfaces, visual environments, etc.). *Vertical frameworks* are developed specifically for a concrete application domain such as telecommunications, manufacturing, multimedia services, etc.

In this paper, the Semantic Web Framework is a horizontal framework that constitutes an abstract reusable design represented by the components commonly involved in the architecture of semantic applications as well as the dependencies between these components.

3 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 [7]. In this context, in which the web is a network of application-usable information, we can define a Semantic Web application as a software application that uses or produces information for the Semantic Web.

As companies begin to perceive the benefits of semantic technologies, they will explore how to apply this technology to build Semantic Web applications. These applications have been characterised by different authors [8] and by events such as the Semantic Web Challenge[1] with the following features:

- Data has semantics and is represented using formal descriptions.
- Semantic data is reused, manipulated and processed.
- Data sources are heterogeneous and are owned or controlled by different organisations.
- Applications assume an open world (i.e., the information is never complete).
- Multiple natural languages are supported.
- RDF(S) and OWL, the open standards recommended by the W3C, are used.

In the Semantic Web, the term reuse appears not only at the data level, as shown above, but also at the application level, because nowadays there exist many open software from a wide range of sources that can be reused when building Semantic Web applications. At the application level, reuse follows three different approaches: a distributed services approach, which integrates web service technology into their architectures; a shared memory approach, which composes components using a shared space of common memory to communicate, as is the case of libraries being reused inside an application; and a mixed approach, which combines the two approaches explained before.

[1] `http://iswc2007.semanticweb.org/callfor/SemanticWebChallenge.asp`

3.1 Semantic Web Application Architectures

Only a few architectures for Semantic Web applications have been proposed so far.

Mika et al. sketch a generic architecture of ontology-based applications grounded in a call-and-return style and structured in hierarchical layers [9]. The layers involved from bottom to top are the following: 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 rests on the ontology and on related services to provide some kind of ontology functionality to an end user.

Tran et al. [10] present a service-oriented architecture also structured in hierarchical layers: the data layer hosts any kind of data sources, including sources different from ontological ones; the logic layer includes application-specific services that are implemented for a particular use case and that operate on specific object models; finally, the presentation layer hosts presentation components that the user interacts with. These authors also classify the components inside the logic layer into ontology services, ontology engineering services and ontology usage services.

By contrast, the framework described in this paper is an open system and is not divided in layers. Layered approaches, on the other hand, present several disadvantages, such as 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 in finding the right level of abstraction, especially if existing systems cross several layers [11].

The two architectures presented above identify some example components that illustrate their approaches. However, in the Semantic Web Framework we have tried to identify exhaustively the existing semantic components of Semantic Web applications. The 32 components we have identified in the Semantic Web Framework cover the 16 and 21 components identified in the previous approaches.

4 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 that state that the Semantic Web Framework should be

- *Developer-oriented*. Different audiences such as developers with low expertise in Semantic Web technologies or ontology practitioners should be considered.
- *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 is 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. have not been taken into account to ease the integration of the components of the Semantic Web Framework into other software architectures.
- *Component based.* To define some specifications of these components that allow 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 by modifying existing ones because the Semantic Web, and its technology, is continuously evolving.

According to the definition of software architecture presented in Section 2, if we want 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 to impose when applying those patterns.

Therefore, this paper is focused on the identification of the components of the Semantic Web Framework; on their classification, as stated below; and on the main interfaces of the Semantic Web Framework 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.

4.1 Definition and Classification of Components

We follow the definition of component given by Szyperski [3] 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. These components can be used either independently or together to develop applications for the Semantic Web; and they can be implemented using services, program libraries or applications.

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 added to the interface that establish use and implementation conditions [3].

Within the Semantic Web Framework, we do not describe the component contracts, since these will be defined in future work, but we explicitly classify the interfaces into the functionalities that a component implements and those that it uses. Therefore, each component is defined by the following:

- *Name.* The name of the component.
- *Description.* A high-level description of the component.

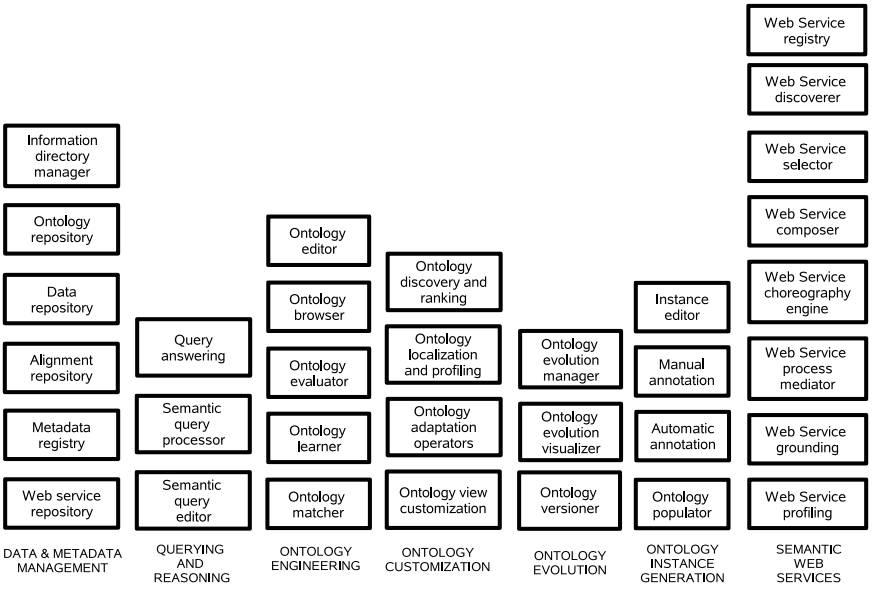

Fig. 1. Components of the Semantic Web Framework

- *Functionalities provided.* An enumeration of the functionalities that the component provides, specifying for each functionality the type or types of interface that it has (user interface, programming interface, service interface, hardware interface, etc.).
- *Component dependencies.* These include an enumeration of the functionalities required by the component to work correctly and that are provided by other components.

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 Semantic Web systems, in other words, the groups of components that provide some specific support to the architecture. These dimensions, however, are not exhaustive; we have classified the different components according to the main functionalities that they provide, as stated in previous Semantic Web technology classifications [12,13].

Figure 1 presents the components that have been identified from software currently available or under construction. The enumeration of components is neither exhaustive nor complete, and is open to improvements and extensions. The current components have been identified by members of the Knowledge Web[2] Network of Excellence who have great expertise in each of the dimensions.

In Figure 1, each dimension of the architecture is represented as a column and reflects those components that provide a particular functionality to the architecture. It should be noted that the order of the components or of the

[2] http://knowledgeweb.semanticweb.org/

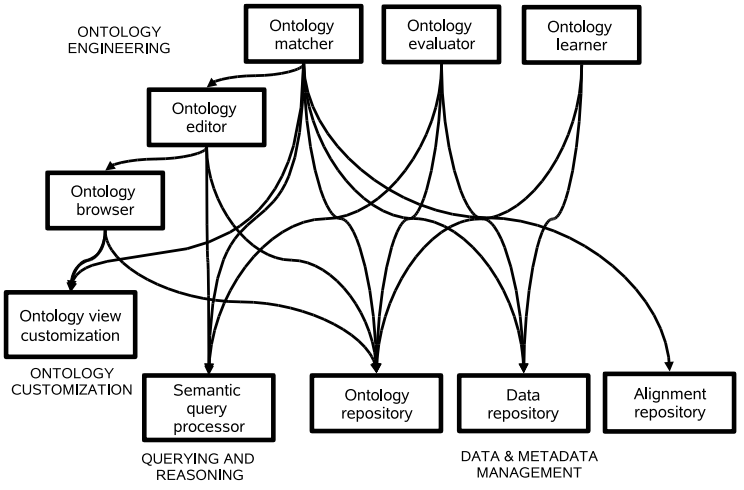

Fig. 2. Dependencies of the components on the *Ontology engineering* dimension

dimensions in the figure does not imply any precedence or relation between them.

The dependencies of each of these components on other components of the framework were identified. Figure 2 shows the basic dependencies of the components on the *Ontology engineering* dimension. Component dependencies are represented graphically in the following way: when one component depends on the functionalities of another, it is then represented with an arrow going from the first component to the component that provides the functionalities.

Existing software that implements the components was also identified. It must be observed that existing implementations may include the functionalities of multiple components. This is clearly seen in ontology engineering platforms, which give support to different tasks of the ontology development process and cover multiple components. In total, we identified 200 component implementations: 43 in the *Data and Metadata Management* dimension, 10 in the *Querying and Reasoning* dimension, 78 in the *Ontology Engineering* dimension, 25 in the *Ontology Customization* dimension, 10 in the *Ontology Evolution* dimension, 15 in the *Ontology Instance Generation* dimension, and 19 in the *Semantic Web Services* dimension.

On the other hand, even if there is a dependency between two components (e.g., an *Ontology editor* requires an *Ontology repository*), in the real world all the implementations of a certain component will not be compatible with all the implementations of the dependent component.

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, dependencies and implementations can be found in [14].

Data and Metadata Management. This dimension includes those components that manage knowledge and data sources, such as:

- *Information directory manager.* It handles query distribution, manages provider directories, identifies information providers from a query, and handles the storage and access to distributed ontologies and data.
- *Ontology repository.* It locally stores and accesses ontologies and instances.
- *Data repository.* It locally stores and accesses data and ontology annotated data.
- *Alignment repository.* It handles the storage and access to distributed alignments.
- *Metadata registry.* It locally stores and accesses metadata information.

Querying and Reasoning. This dimension includes those components that generate and process queries, such as:

- *Query answering.* It takes care of the logical processing of a query by providing reasoning functionalities to search results from a knowledge base.
- *Semantic query processor.* It takes care of the physical processing of a query by providing functionalities to manage query answering over ontologies in distributed sources.
- *Semantic query editor.* It takes care of the user interface for editing queries.

Ontology Engineering. This dimension includes those components that provide functionalities to develop and manage ontologies, such as:

- *Ontology editor.* It allows creating and modifying ontologies, ontology elements, and ontology documentation. These functionalities include single ontology component editing or more advanced editing, such as ontology pruning, extension or specialization.
- *Ontology browser.* It allows visually browsing an ontology.
- *Ontology evaluator.* It evaluates ontologies, either their formal model or their content, during the different phases of the ontology life cycle.
- *Ontology learner.* It acquires knowledge and generates ontologies of a given domain through some kind of (semi)-automatic process.
- *Ontology matcher.* It matches two ontologies or an ontology and another data source and outputs some alignments. Two types of ontology matchers can be distinguished, one that generates matchings and one that uses matchings for other tasks (merging, mediating, etc.).

Ontology Customisation. This dimension includes the components that customize and tailor ontologies, such as:

- *Ontology localization and profiling.* It adapts an ontology according to some context or some user profile.
- *Ontology discovery and ranking.* It finds appropriate views, versions or subsets of ontologies, and ranks them according to some criterion.

- *Ontology adaptation operators.* It is in charge of applying appropriate operators to the ontology in question, resulting in an ontology customized according to some criterion.
- *Ontology view customisation.* It enables the user to change or amend a view on a particular ontology to fit a particular purpose.

Ontology Evolution. This dimension includes those components that manage the ontology evolution, such as:

- *Ontology versioner.* It maintains, stores and manages different versions of an ontology.
- *Ontology evolution visualizer.* It visualises different versions of an ontology.
- *Ontology evolution manager.* It is in charge of the timely adaptation of an ontology to the changes undergone and of the propagation of such changes to dependent artifacts.

Ontology Instance Generation. This dimension includes those components that generate ontology instances, such as:

- *Instance editor.* It allows creating and modifying manually instances of concepts and of relations between such concepts in existing ontologies.
- *Manual annotation.* It allows the manual and the 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).
- *Automatic annotation.* It allows the automatic annotation of digital content (e.g., web pages) with concepts in the ontology. Occurrences in the considered content of concept instances are automatically detected and subsequently annotated.
- *Ontology populator.* It automatically generates new instances in a given ontology from a data source.

Semantic Web Services. This dimension includes those components that discover, select, mediate, compose, choreograph, ground, and profile semantic web services, such as:

- *Web service discoverer.* It publishes and searches service registries, controls access to registries, and distributes and delegates requests to other registries.
- *Web service selector.* After discovering a set of potentially useful services, this component checks whether the services can actually fulfil the user's concrete goal and under what conditions.
- *Web service composer.* It automatically composes web services to provide new value-added web services.
- *Web service choreography engine.* It uses the choreography descriptions of the service requester and provider to drive their conversation.
- *Web service process mediator.* It reconciles the public process heterogeneity that can appear during the invocation of web services.

- *Web service grounding.* It is responsible for web service communication.
- *Web service profiling.* It creates web service profiles based on their execution history.
- *Web service registry.* It registers semantic web services.

5 Use Cases

In order to check the viability of use of the Semantic Web Framework by non-experts from the industry, we selected some of the use cases from Knowledge Web and carried out face-to-face interviews with industry members. Then, a few days before the meeting, we sent them a copy of the Semantic Web Framework specification to read. When the meeting was held, they had the opportunity to raise any questions about the framework they had encountered. Then, their use case was analysed according to the required components. This analysis was led by the industry partner while the Knowledge Web researcher's function was to help the industry partner understand the functioning of the components.

We found out that even before being prompted by the researcher, the industry partners were able to identify most of the components required by their use case and were able to intuitively understand the dependency diagrams, leading to avoidance of inconsistencies (e.g., recognizing that they had forgotten to explicitly add a certain component). In total, 8 use cases were analysed with the Semantic Web Framework. Here we show only one of those use cases, but the reader can find them all in [14].

5.1 Semantic Aggregation of News Stories

We chose a use case from the technology provider Neofonie GmbH[3]. Neofonie represents the typical case of a small company with an interest in deploying semantic solutions to improve their technology offer and better their competitiveness. They have a general knowledge of what semantic technologies are, but lack expert knowledge to successfully evaluate and deploy the technology. We illustrate the framework with their use case as we consider this an ideal scenario for our work to support industry in better modelling of semantic solutions for their needs, the necessary first step before further evaluation and deployment of the technology.

The selected use case deals with the provision of an aggregated news service able to provide business clients with accurate search, thematic clustering, classification of news stories, and e-mail notification of stories of interest. The news sources used are not just the main news feeds and media outlets but also press releases, announcements on websites and other "alternative" sources.

The result of the analysis is shown in Figure 3, which presents the components of the Semantic Web Framework that can support this use case and their dependencies. This analysis could be performed within the company based on a reading of the component descriptions and dependencies, with the final diagram resulting from a briefer meeting with an expert to clarify open issues.

[3] `http://www.neofonie.de`

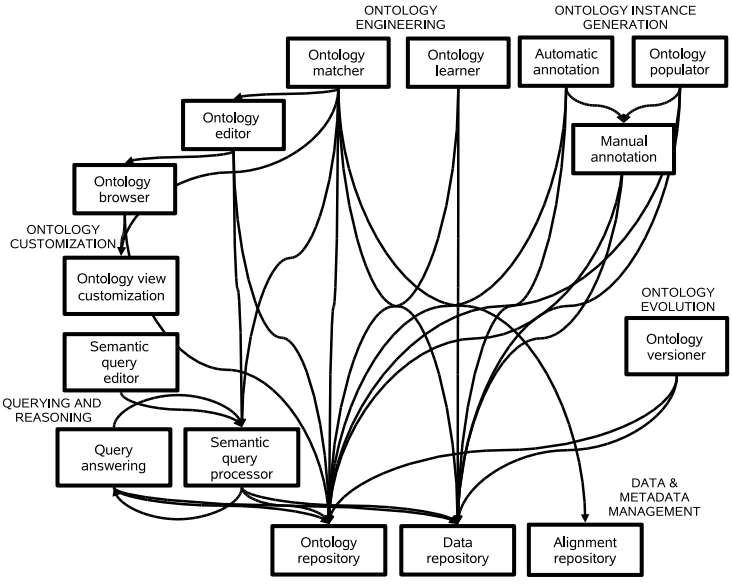

Fig. 3. Components and dependencies for the use case

In order to achieve all of the goals proposed in the business use case, the system could use the following Semantic Web Framework components:

- The *Ontology repository*, the *Data repository*, the *Alignment repository* and the *Metadata registry* store all the data necessary for the use case: the ontologies used for each source, the instance data extracted from these sources and the alignments that have been created between each source ontology.
- The *Query answering*, the *Semantic query processor* and the *Semantic query editor* provide both the user interface support for formulating the query and displaying the results and the system-intern support for performing the query across the aligned instance data and extracting the results.
- The ontologies for representing the data of each source are semi-automatically created using ontology learning techniques through the *Ontology learner* component. The initial ontology extraction is refined with the *Ontology browser* component to view the ontology and the *Ontology editor* component to complete the ontology manually.
- It is possible that with the use of the system over time, the ontologies will need to be revised as new concepts or properties gain relevance. Hence, the *Ontology versioner* component may be employed at a later stage in the system. Likewise, in the ontology extraction part, extracted terms may overlap with those of existing ontologies for related domains such as politics, sport etc. Given the existence of an ontology that represents terms from a certain source, knowledge extraction can take place. Instance data is generated through semi-automatic annotation approaches with the *Automatic annotation* component, the *Manual annotation* component for adding semantic data to news sources, and the *Ontology population* component.

– Finally, two approaches to searching can be considered. In one, queries are expressed in terms of one ontology and, at run time, they are mapped into the other ontologies of the sources; then they are executed across the different source data and the results are combined at the end. However, this approach is very resource intensive at query time. The other approach considered is that, given that we update the source data only periodically, it makes better sense to transform all source data into a core ontology, which can be built from the merge of all source ontologies. Then, we first generate alignments between the source ontologies and a core ontology using the *Ontology matcher* component. These alignments need manual proofing and correction. The alignments also help refine the core ontology. Given now a core ontology and alignments to the individual source ontologies, mediators can be generated for the transformation of instance data from any source in terms of the core ontology. Hence a core ontology is maintained against which the queries are executed.

5.2 Results from Use Cases

The findings of the eight selected use cases reveal that some of the components, namely, the *Ontology repository*, the *Data repository* and the *Metadata registry*, are used in all the use cases. Other components, such as the *Alignment repository*, the *Query answering*, the *Semantic query processor*, the *Ontology editor*, the *Ontology browser*, the *Ontology view customization*, the four components of the *Ontology evolution* dimension, and the *Ontology matcher* are used in almost all the use cases. On the other hand, some other components, namely, the *Information directory manager*, the *Ontology evaluator*, the *Ontology discovery and ranking*, the *Ontology adaptation operators*, the *Instance editor* and all the components of the *Semantic Web Service* dimension are not used in the use cases or almost not used. These findings can serve as an indicator of those fields of research that should be focused on to meet more readily industrial requirements on Semantic Web applications.

Another benefit of this analysis is that the industry members had a basis for choosing which existing Semantic Web tools could be directly re-used in their applications. For each identified component, we provide a list of existing implementations.

Our dependency diagrams are a first step towards a formal analysis of the overall design, where the industry partner can prove whether all dependencies between components are taken into account. In future work, this will be supported further by specifications of component interfaces and reports on component interoperability.

6 Conclusion and Future Work

The Semantic Web Framework is intended to help developers build Semantic Web applications and to diminish development costs. This work is a first step

to provide the foundation for large-scale development of Semantic Web applications; it presents a first definition of the Semantic Web Framework and describes 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 software design phase or the identification of existing implementations of components to be reused. In these cases, having descriptions of the Semantic Web Framework components and their implementations in a machine-processable form can help automate these tasks.

Future work includes providing sets of compatible tools from the components which are already implemented by existing tools. Therefore, the Semantic Web Framework will provide not just single component implementations but also groups of already-interoperable implementations.

We will extend the usability of the framework by providing evaluations and benchmarks of component implementations, interoperability testing between components and cost/benefit models for Semantic Web application development.

Another line of work is to realise the Semantic Web Framework as an infrastructure of semantic focused services so they can be used in the context of a Service Oriented Architecture when semantic functionalities are needed. This will require to develop specifications of the component interfaces, of their interactions, and to develop the middleware needed to adapt the interface specifications to the concrete implementations API. These developments will allow utility computing for semantic resources, i.e., to organise semantic resources so that they may be accessed when needed, just like traditional utilities such as gas, water, or electricity [15].

Within the NeOn project (IST-2005-027595) we are creating a methodology to support the rapid prototyping and development of a new generation of large scale, complex, semantic applications. The overall goal of this methodology is to ensure that economically viable solutions will appear on the market and help application developers to build Semantic Web applications from scratch or by including semantic components into traditional information systems. In this context, the Semantic Web Framework constitutes the starting point of the NeOn methodology that will take into account the existing methods for building component-based software as for example the described in [16].

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, M.C. Suárez-Figueroa, and D. Zyskowski. 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 InnoProfile-Corporate Semantic Web project funded by the German Federal Ministry of Education and Research (BMBF) and the BMBF Innovation Initiative for the New German Länder - Entrepreneurial Regions. Thanks to Rosario Plaza for reviewing the grammar of this paper.

References

1. Oberle, D.: Semantic Management of Middleware. Semantic Web and Beyond (2006)
2. Sommerville, I.: Software Engineering, 8th edn. International Computer Science Series. Addison-Wesley, Reading (2007)
3. Szyperski, C.: Component Software, Beyond Object Oriented Programming. Addison-Wesley, Reading (1998)
4. Krueger, C.W.: Software Reuse. ACM Comput. Surveys 24, 131–183 (1992)
5. IEEE: IEEE Std 1471-2000. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE (2000)
6. Traz, W.: DSSA frequently asked questions. ACM Software Engineering Notes 19, 52–56 (1994)
7. Berners-Lee, T., Handler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
8. Motta, E., Sabou, M.: Next generation semantic web applications. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 24–29. Springer, Heidelberg (2006)
9. Mika, P., Akkermans, H.: D1.2 Analysis of the State-of-the-Art in Ontology-based Knowledge Management. Technical report, SWAP Project (2003)
10. Tran, T., Haase, P., Lewen, H., Muñoz-García, Ó., Gómez-Pérez, A., Studer, R.: Lifecycle-Support in Architectures for Ontology-Based Information Systems. In: Proceedings of the 6th International Semantic Web Conference, pp. 508–522 (2007)
11. Shaw, M., Garlan, D.: Software Architecture: Perspectives on an Emerging Discipline, 1st edn. Prentice Hall, Englewood Cliffs (1996)
12. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering. Springer, Heidelberg (2003)
13. Davies, J., Studer, R., Warren, P. (eds.): Semantic Web Technologies - trends and research in ontology-based systems. John Wiley & Sons, Chichester (2006)
14. García-Castro, R., Muñoz-García, O., Suárez-Figueroa, M., Gómez-Pérez, A., Costache, S., Maynard, D., Dasiopoulou, S., Palma, R., Novacek, V., Lécué, F., Ding, Y., Kaczmarek, M., Piskac, R., Zyskowski, D., Euzenat, J., Dzbor, M., Nixon, L., Léger, A., Vitvar, T., Zaremba, M., Hartmann, J.: D1.2.5 Architecture of the Semantic Web Framework v2. Technical report, Knowledge Web (2007)
15. Pulier, E., Taylor, H.: Understanding Enterprise SOA. Manning (2006)
16. Cheesman, J., Daniels, J.: UML Components. A Simple Process for Specifying Component-Based Software. Component Software Series. Addison-Wesley, Reading (2001)