

Towards a reference architecture for Semantic Web applications

Benjamin Heitmann¹, Conor Hayes¹, and Eyal Oren²

¹ `firstname.lastname@deri.org`
Digital Enterprise Research Institute
National University of Ireland, Galway
Galway, Ireland
² `eyal@cs.vu.nl`
The Network Institute
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands

Abstract The Semantic Web currently has two complimentary architectural approaches: “Bottom-up” emergent best practices by the community and “top-down” prescriptive standards by standards bodies, leaving a gap regarding the concrete implementation of Semantic Web technologies. Based on the Web Science approach of combining empirical analysis with engineering, we are proposing to fill this gap with a reference architecture consisting of seven reusable and domain independent components, extracted from applications submitted at key demonstration challenges in the Semantic Web domain. The reference architecture can be used (1) as a common terminology for communicating concepts and issues related to the implementation of Semantic Web technologies, and (2) as a blueprint when implementing Semantic Web applications.

1 Introduction

Motivation The Semantic Web currently lacks a unified architectural approach. Instead two complimentary approaches are currently evolving, as described in [11, page 68]. The “bottom-up” approach is driven by informal collaborative processes and the emergence of de-facto standards and best practices for sharing data, based on the experience of practitioners. The “top-down” approach is driven by formal standardisation processes, activities and working groups which arrive at standards and ontologies, and which try to establish a prescriptive vision for the Semantic Web.

Between the two approaches a gap opens up regarding the actual implementation of individual applications which leverage Semantic Web standards, which can not be addressed by existing software frameworks for Web applications [15]. Software frameworks and libraries based on an understanding of the architecture of Semantic Web

applications could facilitate the development of Semantic Web applications, but no such architectural foundation exists.

Contributions To address the lack of an architectural foundation for building Semantic Web applications, we are proposing to fill this gap with a reference architecture consisting of seven reusable and domain independent components. Our contributions are: (1) a survey of the functionality of 98 Semantic Web applications, and (2) a reference architecture which can be used to inform and guide the implementation of a Semantic Web application.

Section 2 presents the background for discussing the architecture of the Semantic Web. Section 3 presents the survey results. Section 4 describes the reference architecture, then related work is presented in section 5 and the results of the paper are discussed in section 6.

2 Background

Before we perform our architectural analysis of current Semantic Web applications, we will briefly define the term “architecture”. Since the Semantic Web extends the World-Wide Web [1], we also briefly review existing architectural approaches for Web applications.

Defining software architecture Software architectures are abstractions of runtime systems [7]: an architecture consists of components, connectors and data, and describes patterns for combining the components in a meaningful way; abstracting software systems into architectures is useful when discussing and analysing common behaviour across systems and when developing software frameworks that support application development by providing such common behaviour through off-the-shelf implementations.

The architecture of the World Wide Web The document “Architecture of the World Wide Web, Volume One”[12] represents the result of an iterative process, in which best practices proposed by the community of practitioners and the standards proposed by the World Wide Web consortium informed each other over years. The document describes the overall architecture of the World Wide Web, how multiple standards have to be used together in this architecture, and which community guidelines exist for implementing different aspects of the architecture. A central part of the WWW architecture is Fielding’s Representational State Transfer (REST) architectural style [8], which prescribes a way for networked applications on the Web to interact.

Architectural approaches for the Semantic Web In comparison, the evolution of best practices and standards for Semantic Web technologies has not culminated in a single architectural approach. As argued in [9], the Semantic Web “layer cake” of official standards can be seen as an architecture for the Semantic Web. A second architectural approach is represented by the Linked Data community best practices [2], which leverages the World Wide Web architecture and augments it towards supporting distributed publishing of RDF data. Other best practices e.g. for consistent and persistent naming of URIs [16] are emerging. The W3C task force “Architecture of the World Wide Semantic Web”³ is concerned with providing an approach which encompasses all possible parts of an architecture for the Semantic Web.

Different abstraction levels of architecture Using Fielding’s definition, every abstraction layer of a software system can have its own software architecture. Fielding explicitly differentiates between (a) architecture on the abstraction level of a single application, and (b) architecture on the higher abstraction level of network-based applications, “where the interactions among components are capable of being realised in network communication” [7].

The architecture of individual applications For building Web applications more than 100 application frameworks⁴ provide support. Many frameworks use patterns as the decomposition criterion to arrive at the list of provided components. As described in [13], the predominant pattern used in this context is the model/view/controller pattern, which results in a three tier web application architecture which implements the model through a relational database, the controller through business logic and the view through HTML.

For building applications which leverage Semantic Web standards there is much less support. Semantic Web applications are decentralised and open, operate on distributed data that can be published anywhere, may conform to arbitrary vocabulary and follow semi-structured schemas [15]. Software frameworks for building web applications do not support these characteristics. Different approaches at providing Semantic Web frameworks exist, like the Semantic Web Application Framework (SWAF) for Ruby [14] which provides for accessing an RDF model and creating a customised view. However these are based on specific use cases and not on a general architectural understanding of Semantic Web applications.

³ <http://esw.w3.org/topic/AwwswHome>

⁴ http://en.wikipedia.org/wiki/List_of_web_application_frameworks

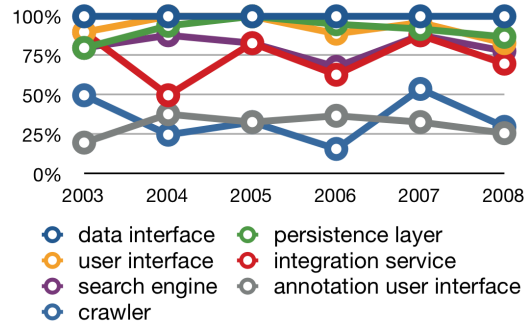


Fig. 1. Trends in implementing the components

3 A survey of Semantic Web applications

The empirical basis for the reference architecture is a survey of 98 applications utilising Semantic Web standards from two key demonstration challenges in the Semantic Web domain: the “Semantic Web challenge”⁵, organised as part of the International Semantic Web Conference from 2003 to 2008, and the “Scripting for the Semantic Web challenge”⁶, organised as part of the European Semantic Web Conference from 2006 to 2008. Duplicate submissions have been eliminated.

Method The components have been extracted from the architecture diagrams and the textual descriptions of the application architecture and implementation, depending on availability in the submitted paper. In a few cases where the decomposition of an application into components was not apparent from a diagram or a description, the similarities in functionality to other applications were used to decompose the application. The results are derived from a checklist of possible functionality, but not verified with the submission authors. The different combinations of components have not been analysed for this survey.

Results The results of the survey are presented in table 1. The results of the survey can also be viewed in more detail on-line⁷.

The surveyed applications share a significant amount of functionality: The (i) data interface provides an abstraction over remote and local data sources, the (ii) persistence layer stores data and run time state, and the (iii) user interface provides access for the user. (i) to (iii) have each been implemented by **more than 90%** of surveyed applications. The (iv) integration service provides a unified view on heterogeneous data, and the (v) search engine al-

⁵ <http://challenge.semanticweb.org/>

⁶ <http://www.semantic scripting.org>

⁷ <http://semwebapp-components.dabbledb.com/>

lows searching in data. (iv) and (v) have each been implemented by **70% to 80%** of surveyed applications. The (vi) crawler discovers and retrieves remote data, and the (vii) annotation user interface allows creating new data. (vi) and (vii) have each been implemented by **30% to 40%** of surveyed applications.

Taken together, each of the components (i) to (v) is implemented by the majority of surveyed applications, giving very strong support for putting them into the reference architecture. Components (vi) and (vii) implement special use cases for Semantic Web technologies.

4 Towards a reference architecture

To define our reference architecture, we abstract from the empirical overview of existing systems, a summary of which was shown in 1. We abstract from the differences between these systems and distinguish seven main components that can be clearly observed to exist in the majority of Semantic Web applications. For each component we will suggest a name, list other common names, give a description of the role and of distinguishing features, and then list the most common variation points of the component found among the surveyed applications.

A **reference architecture** (also known as a reference model) describes high-level concepts and terminology, without fixing interfaces [6, page 242]. It enables discussing the common aspects of implementations from a particular domain, and can be used as a blue print for implementing a single application instance from that domain, which can reduce development and maintenance costs [17].

(i) data interface *Also known as data adapter or data access provider.* Provides the **interface** needed by the application logic to **access local or remote data sources**, with the distinction based on either physical remoteness or administrative and organisational remoteness. Separation from the persistence layer is motivated by the function of the data interface as an **abstraction layer** regarding the implementation, number and distribution of persistence layers.

Possible variation points: Interfacing to local data can be implemented e.g. via native API calls to the persistence layer or SPARQL. Interfacing to remote data can be implemented e.g. via SPARQL or HTTP calls. Federated or distributed access to multiple data sources can be provided. Exporting or importing of e.g. RDF or JSON data might be provided.

(ii) persistence layer: *Also known as persistent store or triple store.* Provides persistent **storage for data and run time state** of the application. Is accessed via the data interface.

Possible variation points: Storage of any combination of structured, semi-structured, unstructured data or (binary) files can be implemented, with different levels of features or optimisation for the different data types. Indexes can enable faster access based on content or structure of data. Possible supported standards include but are not limited to data representation languages (XML, RDF), meta-modelling languages (OWL, RDFS) and query languages (SQL, SPARQL). Inferencing or reasoning based on structure of data or on temporal or spatial features of data may be available.

(iii) user interface: *Also known as portal interface or view.* Provides a **human accessible interface** for using the application and **viewing the data**. Does not provide any capabilities for modifying or creating new data. Separation between the user interface and the annotation user interface is motivated by the low number of applications implementing write access to data.

Possible variation points: The interface can be rendered via HTML and other web standards in a web browser, or via graphics libraries as a native desktop application. The navigation can be based on data or metadata, such as a dynamic menu or faceted navigation. The presentation may be in a generic format, e.g. in a table, or it may use a domain specific visualisation, e.g. on a map, timeline or graph.

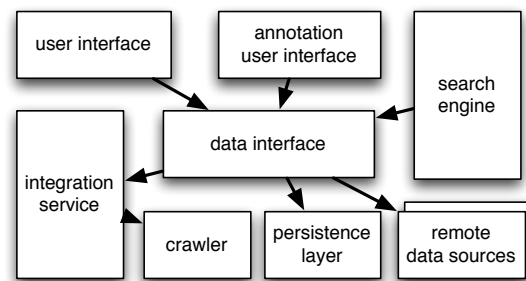


Fig. 2. Example of a hypothetical Semantic Web application implementing the proposed components

(iv) integration service: *Also known as integration, aggregation, mediation, extraction layer or service.* Provides means for **addressing structural, syntactic or se-**

year	number of apps	data interface	persistence layer	user interface	integration service	search engine	annotation user interface	crawler
2003	10	100%	80%	90%	90%	80%	20%	50%
2004	16	100%	94%	100%	50%	88%	38%	25%
2005	6	100%	100%	100%	83%	83%	33%	33%
2006	19	100%	95%	89%	63%	68%	37%	16%
2007	24	100%	92%	96%	88%	88%	33%	54%
2008	23	100%	87%	83%	70%	78%	26%	30%
total	98	100%	91%	92%	72%	81%	32%	35%

Table 1. Percentage of surveyed applications implementing the components, per year and in total

mantic heterogeneity of data, caused by accessing data from multiple data sources using diverse kinds of format, schema or structure. The desired result is a **homogeneous view on all data** for the application. Dependencies to the data interface or the persistence layer can exist, but the integration service provides enough logic for guiding the integration to justify separation from those components.

Possible variation points: Mapping or alignment between different schema may be automatic or manual (with human intervention). Integration may be performed on materialised data or on transient query results. Reasoning, inferencing or custom domain logic may be used for integration. Object consolidation or consistency checking may be part of the integration process. Additional annotations may be generated by the integration process. Integration may be performed once if data stays static or continuously if new data gets added.

(v) search engine: *Also known as query engine or query interface.* Provides the ability to **perform searches** on the data based on the content, structure or domain specific features of the data. **Interfaces for humans, machine agents or both** can be provided.

Possible variation points: Generic full text search or metadata search can be provided. Domain specific customisation of the search can be implemented. Searching may be assisted by refining the search or expanding the search results. The type and number of search results may be explained. The interface for machine agents may be provided by e.g. a SPARQL, web service or REST endpoint.

(vi) crawler: *Also known as harvester, scutter or spider.* Required if data needs to be found and accessed in a domain specific way before it can be integrated. Implements **automatic discovery and retrieval of data**. The low number of applications implementing crawler functionality, and the necessity to crawl before integration data justify this component.

Possible variation points: Support of different discovery and access mechanisms, like HTTP, HTTPS, RSS. Natural language processing or expression matching to parse search results or other web pages can be employed. The crawler can be active once if data is assumed to be static or continuous if new data needs to be discovered.

(vii) annotation user interface: Allows the user to **enter new data, edit existing data, and import or export data**. This component depends on the user interface component, and enhances it with capabilities for modifying and writing data. Separation from the user interface is motivated by the low number of applications implementing write access to data.

Possible variation points: The annotation task can be supported by a dynamic interface based on schema, content or structure of data. Direct editing of data using standards such as e.g. RDF, RDF Schema, OWL or XML can be supported. Input of weakly structured text, using e.g. wiki formatting can be implemented. Suggestions for the user can be based on vocabulary or the structure of the data.

5 Related work

Other surveys about Semantic Web applications are publicly available. [3] presents the results of a survey of 627 Semantic Web researchers and practitioners done in January 2007. The questions from the survey cover the categories of demographics, tools, languages and ontologies.

[4] performs a survey of 35 applications from the “Semantic Web challenges” in 2003, 2004 and 2005. Afterwards the applications are clustered into 24 categories based on their architecture and on their use of metadata. This is then used as the basis to propose a theoretical framework [5] for Semantic Web applications. The result is a prescriptive architecture, whereas our reference architecture aims to provide a descriptive, high level terminology and a standard way of decomposing all Semantic Web applications.

6 Discussion

As the survey shows, there is a large area of functionality that is shared between Semantic Web applications. Our reference architecture proposes a way to partition this functionality, so that it can be better supported by software frameworks and integrated sets of libraries. As discussed in [10], frameworks might have an important role in enabling easy participation and quicker growth of the eco-system which is emerging around the Semantic Web. Without an architectural foundation for such tools, Semantic Web applications will be implemented on a case-by-case basis, due to a missing consensus on required functionality beyond the need to implement Semantic Web standards. Our reference architecture is the foundation for giving software developers ready made building blocks across all areas of typical Semantic Web functionality.

Outlook To complete the proposed architecture, connectors and data flow should be added, as suggested by Fielding's definition of architecture. The prerequisites for the exchangeability of components can be addressed by standardising the interfaces and the data exchange format, e.g. by using RDF. The community grounding of the terminology can be improved with a bigger and more representative sampling of Semantic Web applications.

7 Conclusions

Based on the Web Science approach of combining empirical analysis with engineering, we have presented a reference architecture for Semantic Web applications, consisting of seven reusable and domain independent components: *data interface*, *persistence layer*, *user interface*, *integration service*, *search engine*, *annotation user interface* and *crawler*. The empirical basis of the components is a survey of 98 applications submitted at two key demonstration challenges in the Semantic Web domain.

The immediate benefits of using the reference architecture are: (1) a **common terminology** for communicating concepts and issues related to the implementation of Semantic Web technologies, (2) a **blueprint** to inform and guide the implementation of Semantic Web applications, and (3) a **standard** way of **decomposing** Semantic Web applications.

Taken as a whole, the contributions enable the better understanding of the rapidly evolving landscape of Semantic Web applications.

Acknowledgements The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2). We are grateful for all the comments from R. Cyganiak, M. Dabrowski, A. Harth, M. Hausenblas, J. Jankowski, K. Möller, A. Pas-sant, J. Umbrich.

References

1. T. Berners-Lee. *Weaving the Web*. Collins, 2000.
2. C. Bizer, R. Cyganiak, and T. Heath. How to Publish Linked Data on the Web. Technical report, FU Berlin, 2007.
3. J. Cardoso. The Semantic Web Vision: Where Are We? *IEEE Intelligent Systems*, 22:84–88, 2007.
4. L. M. Cunha and C. J. P. de Lucena. Cluster The Semantic Web Challenges Applications: Architecture and Metadata Overview. Technical report, Pontificia Universidade Catolica do Rio de Janeiro, 2006.
5. L. M. Cunha and C. J. P. de Lucena. A Semantic Web Application Framework. Technical report, Pontificia Universidade Catolica do Rio de Janeiro, 2007.
6. A. Endres and D. Rombach. *A Handbook of Software and Systems Engineering*. Pearson Education, 2003.
7. R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, UC Irvine, 2000.
8. R. T. Fielding and R. N. Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology*, 2002.
9. A. Gerber, A. van der Merwe, and A. Barnard. A Functional Semantic Web Architecture. *Proceedings of the European Semantic Web Conference*, 2008.
10. B. Heitmann. Transitioning web application frameworks towards the semantic web. Master's thesis, Universität Karlsruhe, 2007.
11. J. Hendler, N. Shadbolt, W. Hall, T. Berners-Lee, and D. Weitzner. Web science: an interdisciplinary approach to understanding the web. *Communications of the ACM*, 51(7):60–69, 2008.
12. I. Jacobs and N. Walsh. Architecture of the world wide web, volume one. W3C recommendation, W3C, Dec. 2004.
13. A. Leff and J. Rayfield. Web-application development using the model/view/controller design pattern. *Proceedings of the International Enterprise Distributed Object Computing Conference*, pages 118–127, 2001.
14. E. Oren, A. Haller, M. Hauswirth, B. Heitmann, S. Decker, and C. Mesnage. A flexible integration framework for semantic web 2.0 applications. *Software, IEEE*, 2007.
15. E. Oren, B. Heitmann, and S. Decker. ActiveRDF: embedding Semantic Web data into object-oriented languages. *Journal of Web Semantics*, 2008.
16. L. Sauer mann, R. Cyganiak, and M. Volkel. Cool URIs for the Semantic Web. W3C note, W3C, 2008.
17. M. Shaw and D. Garlan. *Software Architecture. Perspectives of an Emerging Discipline*. Prentice Hall, 1996.