An empirically-grounded conceptual architecture for applications on the Web of Data

Benjamin Heitmann, Richard Cyganiak, Conor Hayes, and Stefan Decker

***Abstract*—We present a component-based, conceptual architecture for Semantic Web applications. It describes the high-level functionality which differentiates applications using RDF from database-driven applications. We provide a strong empirical grounding for this architecture through a survey of Semantic Web applications over most of the past decade. Our empirical approach allows us to describe the current state of the art for developing and deploying applications on the Web of Data. In addition, we determine how far the adoption of signature research topics of the Semantic Web, such as data reuse, data integration and reasoning, has progressed. We then discuss the main implementation challenges facing developers using Semantic technologies, as observed in the survey. We build on this in order to suggest future approaches to facilitate the standardisation of components and the development of software engineering tools to increase uptake of the Web of Data.**

***Index Terms*—Semantic Web, Web of Data, Software Engineering, empirical survey, conceptual architecture.**

I. Introduction

TO date, investigating Semantic Web technologies and the Web of Data in terms of their cost of implementation complexity has not been given the same priority as the research on potential benefits. We argue that the lack of emphasis on simplifying the development and deployment of Semantic Web-enabled applications has been an obstacle for real-world adoption of Semantic Web technologies and the emerging Web of Data.

Semantic Web technologies simplify knowledge-intensive applications by enabling a Web of interoperable and machine-readable data [1] based on formal and explicit descriptions of the structure and semantics of the data [2]. This emerging new Web is referred to as the Web of Data [3]. Up to now, Web applications have been designed around relational database standards of data representation and service. A move to an RDF-based data representation introduces challenges for the application developer in rethinking the Web application outside the standards and processes of database-driven Web development. These include, but are not limited to, the graph-based data model of RDF [2], the Linked Data principles [4], and formal and domain specific semantics [5].

In this article we perform an *empirical survey* of Semantic Web applications over most of the past decade. This allows us to describe the current state of the art for developing and deploying Semantic Web applications. It also allows us to determine how far the adoption of signature research topics of the Semantic Web, such as data reuse, data integration and reasoning, has progressed.

We then use the results of this survey as the empirical foundation for a component-based *conceptual architecture* for Semantic Web applications. Our conceptual architecture describes the high level components which are most often observed among the participants of the survey. These components implement the functionality that differentiates applications using RDF from database-driven applications. For each component we describe the most common implementation strategies that are suited to specific application goals and deployment scenarios. The existence of such an architecture can be seen as evidence of a convergence to a common set of principles to solve recurring development challenges.

Based on the empirical survey and the conceptual architecture, we discuss the main *implementation challenges* facing developers using Semantic technologies. In addition we suggest *future approaches* in using the conceptual architecture to facilitate the standardisation of components and the development of software engineering tools to increase the uptake of the Web of Data.

As shown in table I, the article presents this perspective through the following sections: Section II introduces the different terms related to the Web of Data and the Semantic Web. Section III presents the findings of the empirical survey, including our research method. We also discuss potential counter-arguments to the validity of our results. Based on this empirical foundation, section IV proposes a conceptual architecture for Semantic Web applications, and discusses

1.	Empirical foundation: survey of 124 Semantic Web applications	Section III
2.	Conceptual architecture for Semantic Web applications (empirically-grounded in survey)	Section IV
3.	Main challenges for implementing Semantic Web technologies (as observed in the survey)	Section IV-D
4.	Future approaches for applying the conceptual architecture	Section IV-E
5.	Related work: surveys about, and architectures for Semantic Web applications	Section V

TABLE I
STRUCTURE OF THIS ARTICLE

B. Heitmann, R. Cyganiak, C. Hayes and S. Decker are with the Digital Enterprise Research Institute (DERI), NUI Galway, Ireland. (e-mail: benjamin.heitmann@deri.org, richard.cyganiak@deri.org, conor.hayes@deri.org, stefan.decker@deri.org)

The work presented in this paper has been funded by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Líon-2).

Manuscript received June 1, 2010; revised November 1, 2010.

the main implementation challenges presented by Semantic Web technologies. We present several approaches to ease the development of Semantic Web applications in the future. Finally, section V discusses related work, and section VI concludes the article and presents future work.

II. Background

The term *Web of Data* refers to a subset of the World Wide Web. While the World Wide Web provides the means for creating a web of human readable documents, the Web of Data aims to create a web of structured, machine-readable data [1] based on formal and explicit descriptions of the structure and semantics of the data [2]. The availability of such data facilitates the creation of knowledge-intensive applications, potentially simplifying data integration [6] and data aggregation [5].

In order to build a single Web of Data, all data providers have to follow shared guidelines for publishing their data and connecting it to other data sources. To date, the set of guidelines that has attracted the largest mind-share, and achieved a critical mass of supporters, are the *Linked Data principles* [4]. These principles describe how to integrate a subset of the *Semantic Web* technology stack[1] with the existing standards from the World Wide Web, in order to enable the Web of Data to be a true subset of the Web.

The Semantic Web technologies used for Linked Data are: the Resource Description Framework (RDF), which provides a graph based data model and the basic, domain-independent formal semantics for the data model; the SPARQL Query Language that allows querying RDF data with graph patterns, and provides basic means for transforming RDF data between different schemata. In addition, technologies from the World Wide Web provide the fundamental infrastructure: Uniform Resource Identifiers (URIs) are used to provide globally unique identifiers for the data, and the HyperText Transfer Protocol (HTTP) is used for accessing and transporting the data.

The Linked Data principles have been adopted by an increasing number of data providers, especially from the *Linking Open Data* (LOD) community project, which makes free and public data available as Linked Data. As of September 2010[2] this includes 203 data sets with over 25 billion RDF triples, which are interlinked by around 395 million RDF links.

III. An empirical survey of Semantic Web applications

The evolution of Semantic Web technologies is characterised by the constant introduction of new ideas, standards and technologies, and thus appears to be in a permanent state of flux. However, nearly a decade has passed since Tim Berners-Lee et al. published their comprehensive vision for a Semantic Web [1] in 2001. This allows us to look back on the empirical evidence left behind by the Semantic Web applications that have been developed and deployed during this time.

Sjoberg et. al [7] argue that there are relatively few empirical studies in software engineering research, and that the priority for evaluating new technologies is lower than that of developing new technologies. We can transfer this observation to research on the engineering of Semantic Web applications. Without empirical evidence it is difficult to evaluate the adoption rate of Semantic Web technologies. To provide such insights, we collect evidence from a representative sample of the population of developers deploying Semantic Web technologies. In particular we perform an empirical survey of over a hundred Semantic Web applications from two demonstration challenges that received much attention from the Semantic Web research community.

The results of our survey enable us to determine the state of adoption of key Semantic Web research goals such as data reuse, data integration and reasoning. In this section, we first explain our research method. Then the findings of our empirical survey are presented, after which we discuss threats to the validity of the survey.

A. Research method of the survey

The empirical evidence for our survey is provided by 124 applications that were submitted to two key demonstration challenges in the Semantic Web domain: (1) the "Semantic Web challenge"[3] [8] in the years 2003 to 2009 with 101 applications, which is organised as part of the International Semantic Web Conference; and (2) the "Scripting for the Semantic Web challenge"[4] in the years 2006 to 2009 with 23 applications, which is organised as part of the European Semantic Web Conference. As the Semantic Web was an emerging research topic, the majority of applicants were from research centres or university departments, though there were some industrial participants.

Each challenge awards prizes for the top entries, as selected by the judges of the respective contest. In order to apply, the developers of an application were required to provide the judges with access to a working instance of their application. In addition, a scientific paper describing the goals and the implementation of the application was also required.

The applications which were submitted to the "Semantic Web challenge" had to fulfill a set of minimal requirements, as described in [8]: The information sources of the application had to be geographically distributed, should have had diverse ownership so that there was no control of the evolution of the data; the data was required to be heterogeneous and from a real-world use-case. The applications should have assumed an open world model, meaning that the information never was complete. Lastly the applications were required to use a formal description of the data's meaning. Applications submitted to the "Scripting for the Semantic Web challenge" only had to be implemented using a scripting programming language.

We collected the data about each application through a questionnaire that covered the details about the way in which each application implemented Semantic Web technologies and standards. It consisted of 12 questions which covered

[1] http://en.wikipedia.org/wiki/Semantic_Web_Stack
[2] http://lod-cloud.net/
[3] http://challenge.semanticweb.org/
[4] http://www.semanticscripting.org

	2003	2004	2005	2006	2007	2008	2009	overall
Programming languages	Java 60% C 20%	Java 56% JS 12%	Java 66%	Java 10% JS 15% PHP 26%	Java 50% PHP 25%	Java 43% PHP 21%	Java 46% JS 23% PHP 23%	Java 48% PHP 19% JS 13%
RDF libraries	—	Jena 18% Sesame 12% Lucene 18%	—	RAP 15% RDFLib 10%	Sesame 33% Jena 8%	Sesame 17% ARC 17% Jena 13%	Sesame 23%	Sesame 19% Jena 9%
SemWeb standards	RDF 100% OWL 30%	RDF 87% RDFS 37% OWL 37%	RDF 66% OWL 66% RDFS 50%	RDF 89% OWL 42% SPARQL 15%	RDF 100% SPARQL 50% OWL 41%	RDF 100% SPARQL 17% OWL 10%	RDF 100% SPARQL 69% OWL 46%	RDF 96% OWL 43% SPARQL 41%
Schemas/ vocabularies/ ontologies	RSS 20% FOAF 20% DC 20%	DC 12% SWRC 12%	—	FOAF 26% RSS 15% Bibtex 10%	FOAF 41% DC 20% SIOC 20%	FOAF 30% DC 21% DBpedia 13%	FOAF 34% DC 15% SKOS 15%	FOAF 27% DC 13% SIOC 7%

TABLE II
IMPLEMENTATION DETAILS BY YEAR, TOP 3 ENTRIES PER CELL IF POSSIBLE. ENTRIES WITH JUST ONE APPLICATION ARE OMITTED

	2003	2004	2005	2006	2007	2008	2009
manual	30%	13%	0%	16%	9%	5%	4%
semi-automatic	70%	31%	100%	47%	58%	65%	61%
automatic	0%	25%	0%	11%	13%	4%	19%
not needed	0%	31%	0%	26%	20%	26%	16%

TABLE III
DATA INTEGRATION BY IMPLEMENTATION STRATEGY AND YEAR

	2003	2004	2005	2006	2007	2008	2009
Data creation	20%	37%	50%	52%	37%	52%	76%
Data import	70%	50%	83%	52%	70%	86%	73%
Data export	70%	56%	83%	68%	79%	86%	73%
Inferencing	60%	68%	83%	57%	79%	52%	42%
Decentralised sources	90%	75%	100%	57%	41%	95%	96%
Multiple owners	90%	93%	100%	89%	83%	91%	88%
Heterogeneous formats	90%	87%	100%	89%	87%	78%	88%
Data updates	90%	75%	83%	78%	45%	73%	50%
Linked Data principles	0%	0%	0%	5%	25%	26%	65%

TABLE IV
RESULTS OF THE BINARY PROPERTIES FROM THE APPLICATION QUESTIONNAIRE

the following areas: (1) usage of programming languages, RDF libraries, Semantic Web standards, schemas, vocabularies and ontologies; (2) data reuse capabilities for data import, export or authoring; (3) implementation of data integration and schema alignment; (4) use of inferencing; (5) data access capabilities for the usage of decentralised sources, usage of data with multiple owners, data with heterogeneous formats, data updates and adherence to the Linked Data principles. The full data of the survey and a description of all the questions and possible answers is available online[5].

For each application the data was collected in two steps: Firstly, the application details were filled into the questionnaire based on our own analysis of the paper submitted with the application. Then we contacted the authors of each paper, and asked them to verify or correct the data about their application. This allowed us to fill in questions about aspects of an application that might not have been covered on a paper. For instance the implementation details of the data integration are not discussed by most papers. 65% of authors replied to this request for validation.

[5]http://the-blank.net/semwebappsurvey/

B. Findings

The results of our empirical survey show that the adoption of the capabilities that characterise Semantic Web technologies has steadily increased over the course of the demonstration challenges. In this section we present the results of the survey and analyse the most salient trends in the data. Tables II, III and IV summarise the main findings.

Table II shows the survey results about the programming languages and RDF libraries that were used to implement the surveyed applications. In addition, it shows the most implemented Semantic Web standards and the most supported schemas, vocabularies and ontologies. Java was the most popular programming language choice throughout the survey time-span. This accords with the fact that the two most mature and popular RDF libraries, Jena and Sesame both require Java. On the side of the scripting languages, the most mature RDF library is ARC, which explains the popularity of PHP for scripting Semantic Web applications. From 2006 there is a noticeable consolidation trend towards supporting RDF, OWL and SPARQL, reflecting an emergent community consensus. The survey data about vocabulary support requires a different interpretation: While in 2009 the support for the top three vocabularies went down, the total number of supported vocabularies went from 9 in 2003 to 19 in 2009. This can be explained by the diversity of domains for which data became available as the Web of Data expanded.

The simplification of data integration is claimed as one of the central benefits of implementing Semantic Web technologies. Table III shows the implementation of data integration grouped by implementation strategy and year. The results suggest that, for a majority of applications, data integration still requires manual inspection of the data and human creation

of rules, scripts and other means of integration. However the number of applications that implement fully automatic integration is on the rise, while the number of applications which require manual integration of the data is steadily declining. The steady number of applications that do not require data integration, can be explained by the availability of homogeneous data sources from the Web of Data that do not need to be integrated.

Table IV shows the survey results on the support for particular features over the time period. Some capabilities have been steadily supported by a majority of applications throughout the whole period, while other capabilities have seen increases recently or have suffered decreases.

Support for decentralised sources, data from multiple owners, and sources with heterogeneous formats is supported each year by a large majority of the applications. We can attribute this to the central role that these capabilities have in all of the early foundational standards of the Semantic Web such as RDF and OWL. While support for importing and exporting data has fluctuated over the years, it still remains a popular feature. Since 2007, the SPARQL standard has been increasingly used to implement APIs for importing data.

The Linked Data principles are supported by more then half of the contest entries in 2009, after being formulated in 2006 by Tim Berners-Lee [4]. Interestingly the increasing usage of the Linking Open Data (LOD) may explain some of the positive and negative trends we observe in other capabilities. For example, the negative correlation (-0.61) between support for linked data principles and inferencing is probably explained by the fact that data from the LOD cloud already uses RDF and usually does not require any integration.

Support for the creation of new structured data is strongly correlated (0.75) with support for the Linked Data principles. This can be explained by the growing maturity of RDF stores and their APIs for creating data, as well as increased community support for adding to the Linked Data cloud. On the other hand, support for updating data after it was acquired is strongly negatively correlated (-0.74) with the growth in support for the LOD principles. This could reflect a tendency to treat structured data as being static.

To summarise: Support for features which differentiate RDF-based from database-driven applications is now widespread amongst the development communities of the challenges. Support for the graph-based data model of RDF is virtually at 100%, and most applications reuse formal domain semantics such as the FOAF, DublinCore and SIOC vocabularies. Support for the Linked Data principles has reached more then half of the contest entries in 2009. And while the choice of implemented standards has crystallised around RDF, OWL and SPARQL, the choice of programming languages and RDF libraries has gravitated towards Java, Sesame and Jena, as well as PHP and ARC. In addition, the majority of applications still require some human intervention for the integration service.

C. Threats to validity

The first threat to validity concerns the selection of applications for our empirical survey. We choose to use only applications from two academic demonstration challenges for our survey. This may raise the question of representativeness, as most of the applications are academic prototypes. However, as both challenges explicitly also invited and received some industry submissions, the two challenges do not have a purely academic audience.

Furthermore, we believe that the data of our survey is representative for the following reasons: Firstly, the applications could not just be tools or libraries, but had to be complete applications that demonstrate the benefits of the Web of Data to the end-user of the application. Secondly, the applications submitted to the challenges already follow baseline criteria as required in [8], e.g. most of them use real-world data. Thirdly, the authors of each submitted application were also required to submit a scientific paper focusing on the implementation aspects of their application, which allows them to explain the goals of their application in their own terms. Finally, each challenge was held over multiple years, which makes it easier to compare the submissions from the different years in order to see trends, such as the uptake of the Linked Data principles.

The second threat to validity, is the number of authors who verified the data about their applications. Only 65% of authors replied to this request for validation. This may be due to the short-lived nature of academic email addresses. In several instances, we tried to find an alternative email address, if an address had expired. Every author that we successfully contacted validated the data about their application, which usually included only small corrections for one or two properties. We were unable to validate the data for authors that could not be reached. As the first challenge was already held in 2003, we do not believe that a much higher validation rate would have been possible at the time of writing.

The third threat to validity is that for each application only the associated academic paper and not the source code were analysed for the survey. There are several reasons for this. At the time of writing most demonstration prototypes, except those from the most recent years, were not deployed anymore. Publication of the source code of the applications was not a requirement for both challenges. In addition, it is very likely that most applications would not have been able to make their source code available due to IP or funding restrictions.

IV. Empirically-grounded conceptual architecture

Based on our empirical analysis, we now present a conceptual architecture for Semantic Web applications. As defined by Soni et al. [9], a *conceptual architecture* describes a system in terms of its major design elements and the relationships among them, using design elements and relationships specific to the domain. It enables the discussion of the common aspects of implementations from a particular domain, and can be used as a high-level architectural guideline when implementing a single application instance for that domain.

Our conceptual architecture describes the high level components most often used among the surveyed applications to implement functionality that substantially differentiates RDF-supported applications from database-driven applications. For

year	number of applications	graph access layer	RDF store	graph-based navigation interface	data homogenisation service	graph query language service	structured data authoring interface	data discovery service
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%
2009	26	100%	77%	88%	80%	65%	19%	15%
total	124	100%	88%	91%	74%	77%	29%	30%

TABLE V
RESULTS OF THE ARCHITECTURAL ANALYSIS BY YEAR AND COMPONENT

each component we describe the most common implementation strategies that are suited for specific application goals and deployment scenarios.

We first explain the choice of architectural style for our conceptual architecture, and we describe our criteria for decomposing the surveyed applications into components. Then we provide a detailed description of the components. This is followed by a discussion of the main implementation challenges that have emerged from the survey data. In addition, we suggest future approaches for applying the conceptual architecture.

A. Architectural style of the conceptual architecture

Fielding [10] defines an architectural style as a coordinated set of architectural constraints that restricts the roles and features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style.

Our empirical survey of Semantic Web applications showed that there is a range of architectural styles among the surveyed applications, which is dependent on the constraints of the respective software architects. For example, as existing Web service infrastructure can be reused as part of the Semantic Web application, many applications chose a *service-oriented architecture*. Applications that are focused on reusing existing databases and middleware infrastructure preferred a *layered style*. The *client-server architecture* was used when decentralised deployment across organisations or centralised storage were important. Applications that prioritised robustness, decentralisation or independent evolution favoured a *peer-to-peer architecture*. Finally, the architecture of all other applications usually was defined by the reuse of existing tools and libraries as components in a *component-based architecture*.

For this reason we chose the component-based architectural style for our conceptual architecture. It allows us to express the most common high-level functionality that is required to implement Semantic Web technologies as separate components.

B. Decomposing the surveyed applications into components

In order to arrive at our component-based conceptual architecture, we started with the most commonly found functionality amongst the surveyed applications: a component that handles RDF, an integration component and a user interface. With these components as a starting point, we examined whether the data from the architectural analysis suggested we split them into further components. Table V shows the results of this architectural analysis. The full data of the analysis is available online[6].

Every surveyed application (100%) makes use of RDF data. In addition, a large majority (88%), but not all surveyed applications implement persistent storage for the RDF data. In practice many triple stores and RDF libraries provide both functionality, but there are enough cases where this functionality is de-coupled, e.g. if the application has no local data storage, or only uses SPARQL to access remote data. This requires splitting of the RDF-handling component into two components. First, a *graph access layer*, which provides an abstraction for the implementation, number and distribution of data sources of an application. Second, an *RDF store* for persistent storage of graph-based RDF data.

A query service which implements SPARQL, or other graph-based query languages, in addition to searching on unstructured data, is implemented by 77% of surveyed applications. Such high coverage suggested the need for a *graph query language service*. It is separate from the graph access layer, which provides native API access to RDF graphs.

A component for integrating data is implemented by 74% of the applications. It provides a homogeneous perspective on the external data sources provided by the graph access layer. Usually external data first needs to be discovered and aggregated before it can be integrated - an integration service would offer this functionality. However only 30% of the surveyed applications required this functionality. For this reason, we split the integration functionality into a *data homogenisation service* and a *data discovery service*.

Most (91%) applications have a user interface. All user interfaces from the surveyed applications allow the user to navigate the graph-based data provided by the application. Only 29% provide the means for authoring new data. Thus the user interface functions were split between two components: the *graph-based navigation interface* and the *structured data authoring interface*.

C. Description of components

The resulting seven components describe the largest common high-level functionality which is shared between the surveyed applications in order to implement Semantic Web technologies. For each component we give a name, a description of the role of the component, related research references, and a list of the most common implementation strategies for each component. Table VI shows all of the components and connectors, figure 1 shows an example component diagram using these components and connectors.

1) Graph access layer: This component provides the interface needed by the application logic to access local or remote data sources, with the distinction based on physical, administrative or organisational remoteness. In addition this

[6]http://preview.tinyurl.com/component-survey-results-csv

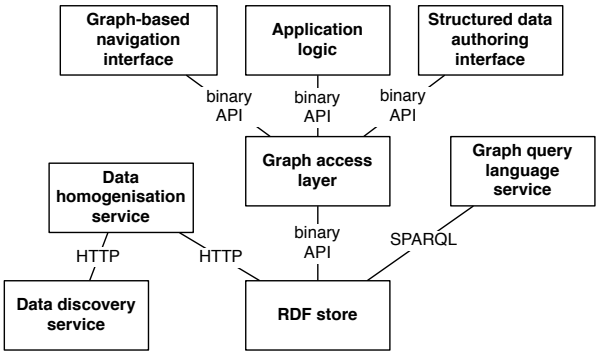

Fig. 1. Example of a component diagram using the components and connectors of our conceptual architecture

Components	Connectors
Graph access layer	HTTP
RDF store	HTTPS
Data homogenisation service	SPARQL over HTTP
Data discovery service	SQL connection
Graph query language service	binary API
Graph-based navigation interface	
Structured data authoring interface	

TABLE VI
SUMMARY OF COMPONENTS AND CONNECTORS

component provides a translation or mapping from the native data model of the programming language to the graph-based data model of RDF. All (100%) of the applications have a graph access layer. [11] provides a detailed overview of different libraries for accessing RDF data.

Implementation strategies: Accessing local data is implemented via programmatic access through RDF libraries by at least 50% of the applications. 47% use a query language for accessing local or remote data sources. Most of these applications use the SPARQL standard. Decentralised sources are used by 85%; 91% use data from multiple owners; 73% can import external data and 69% can export their data or provide a SPARQL end-point to make their data reusable. 65% of applications support data-updating during run-time.

2) RDF store: This component provides persistent storage for RDF and other graph based data. It is accessed via the graph access layer. 88% of applications have such functionality. The most popular solutions for persistent storage of RDF data are provided by Sesame [12] and Jena [13].

Implementation strategies: Possible supported standards include, but are not limited to, data representation languages (XML, RDF), meta-modelling languages (OWL, RDFS) and query languages (SQL, SPARQL). RDF is explicitly mentioned by 96% of applications; OWL is supported by 43%; RDF Schema is supported by 20%. Inferencing or reasoning on the stored data is explicitly mentioned by 58% of the applications. Another implementation strategy is to use a relational database to store RDF data.

3) Data homogenisation service: This component provides a means for addressing the structural, syntactic or semantic heterogeneity of data resulting from data access to multiple data sources with different formats, schemas or structure. The service goal is to produce a homogeneous view on all data for the application. The data homogenisation service often needs to implement domain or application specific logic. 74% of the applications implement this component. [14] provides a general overview of semantic integration, and [15] provides an overview of using Semantic Web technologies for integration.

Implementation strategies: Integration of heterogeneous data is supported by 87% of the applications; 91% support data integration from sources with different ownership. Integration of data from distributed, decentralised data sources is supported by 85%. These three properties are orthogonal, as it would be possible, for example, to support just SIOC data, which is not heterogeneous, but which may be aggregated from personal websites, so that the data sources are distributed and under different ownership.

The four major styles of implementing the data homogenisation service are: Automatic integration (11%), which is performed using heuristics or other techniques to avoid human interaction. Semi-automatic integration (59%), which requires human inspection of the source data, after which rules, scripts and other means are manually created to integrate the data automatically in the future. Manual integration (10%) in which the data is completely edited by a human. Finally, 20% do not need any data integration because they operate on homogeneous sources.

4) Data discovery service: This component implements automatic discovery and retrieval of external data. This is required where data should be found and accessed in a domain specific way before it can be integrated. 30% of applications implement a data discovery service. [16] introduces different research issues related to data crawling, and [17] provides an empirical survey of the change frequency of Linked Data sources.

Implementation strategies: The component should support 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 service can run once if data is assumed to be static or continuously if the application supports updates to its data (65%).

5) Graph query language service: This component provides the ability to perform graph-based queries on the data, in addition to search on unstructured data. Interfaces for humans, machine agents or both can be provided. 77% of applications provide a graph query language service. [18] provides a survey and classification of semantic search approaches, and [19] provides an analysis of approaches for user interaction with semantic search services.

Implementation strategies: Besides search on features of the data structure or semantics, generic full text search can be provided. An interface for machine agents may be provided by a SPARQL, web service or REST endpoint. SPARQL is implemented by 41% of applications, while the preceding standards SeRQL and RDQL are explicitly mentioned by only 6%.

6) Graph-based navigation interface: This component provides a human accessible interface for navigating the graph-based data of the application. It does not provide any capabilities for modifying or creating new data. 91% of applications have a graph-based navigation interface. Faceted browsing is a part of many user interfaces for structured data [20]. One of the most popular JavaScript libraries for displaying RDF and other structured data as faceted lists, maps and timelines is Exhibit [21]. The general role of user interfaces for semantic portals is described in [22].

Implementation strategies: 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.

7) Structured data authoring interface: This component allows the user to enter new data, edit existing data, and import or export data. The structured data authoring component depends on the navigation interface component, and enhances it with capabilities for modifying and writing data. Separation between the navigation interface and the authoring interface reflects the low number of applications (29%) implementing write access to data. The Semantic MediaWiki project [23] is an example where a user interface is provided for authoring data.

Implementation strategies: 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, for example, wiki formatting can be implemented. Suggestions for the user can be based on vocabulary or the structure of the data.

D. Implementation challenges

When comparing the development of database-driven and RDF-based applications, different implementation challenges arise that are unique to Semantic Web technologies. Based on our empirical analysis we have identified the most visible four challenges: Integrating noisy data, mismatched data models between components, distribution of application logic across components and missing guidelines and patterns.

Identifying these challenges has two benefits. It allows practitioners to better estimate the effort of implementing Semantic Web technologies. In addition, it allows tool developers to anticipate and mitigate these challenges in future versions of new libraries and software frameworks.

1) Integrating noisy and heterogeneous data: One objective of RDF is to facilitate data integration [6] and data aggregation [5]. However, the empirical data from our analysis shows that the integration of noisy and heterogeneous data contributes a major part of the functional requirements for utilising Semantic Web technologies and data.

Our analysis demonstrates the impact of the integration challenge on application development: The majority (74%) of analysed applications implemented as a data homogenisation service. However some degree of manual intervention was necessary for 69% of applications. This means that prior to integration, data was either manually edited or data from the different sources was inspected in order to create custom rules or code. Only 11% explicitly mention fully automatic integration using e.g. heuristics or natural language processing. 65% allowed updating of the data after the initial integration, and reasoning and inferencing was also used for 52% of integration services.

2) Mismatch of data models between components: The surveyed applications frequently had to cope with a software engineering mismatch between the internal data models of components. Accessing RDF data from a component requires mapping of an RDF graph to the data model used by the component [24].

Most of the analysed applications (about 90%) were implemented using object-oriented languages, and many of the analysed applications stored data in relational databases. This implies that these applications often had to handle the mismatch between three different data-models (graph-based, relational and object-oriented). This can have several disadvantages, such as introducing a high overhead in communication between components and inconsistencies for round-trip conversion between data models.

3) Distribution of application logic across components: For many of the components identified by our analysis, the application logic was not expressed as code but as part of queries, rules and formal vocabularies. 52% of applications used inferencing and reasoning, which often encode some form of domain and application logic; the majority of applications explicitly mentioned using a formal vocabulary, and 41% make use of an RDF query language. This results in the application logic being distributed across the different components.

The distribution of application logic is a well known problem for database-driven Web applications [25]. Current web frameworks such as Ruby on Rails[7] or the Google Web Toolkit[8] allow the application developer to control the application interface and the persistent storage of data programmatically through Java or Ruby, without resorting to a scripting language for the interface and SQL for the data storage. Similar approaches for centralising the application logic of Semantic Web applications still have to be developed.

4) Missing guidelines and patterns: Most Semantic Web technologies are standardised without providing explicit guidelines for the implementation of the standards. This can lead to incompatible implementations especially when the interplay of multiple standards is involved. The establishment of a community consensus generally requires comparison and alignment of existing implementations. However, the waiting period until agreed guidelines and patterns for the standard have been published can have a negative impact on the adoption of a standard. To illustrate this implementation challenge, we explain the most visible instances from our survey data.

Publishing and consuming data: All of the applications consume RDF data of some form; 73% allow access to or

[7] http://rubyonrails.org/

[8] http://code.google.com/webtoolkit/

importing of user-provided external data, and 69% can export data or can be reused as a source for another application. However the analysis shows that there are several incompatible possible implementations for publishing and consuming of RDF data. Only after the publication of the Linked Data principles [4] in 2006 and the best practices for publishing Linked Data [26] in 2007, did an interoperable way for consuming and publishing of data emerge. This is also reflected by the increasing use of these guidelines by the analysed applications from 2007 on.

Embedding RDF on web pages: While the majority of applications in our survey have web pages as user interfaces, RDF data was published or stored separately from human readable content. This resulted in out-of-sync data, incompatibilities and difficulties for automatically discovering data. Our analysis shows, that after finalising the RDFa standard in 2008, embedding of RDF on web pages strongly increased.

Writing data to a remote store: While SPARQL standardised remote querying of RDF stores, it did not include capabilities for updating data. Together with a lack of other standards for writing or updating data, this has resulted in a lot of applications (71%) which do not include a user interface for authoring data. This is being addressed in the next versions of SPARQL which will include data editing and updating capabilities[9].

Restricting read and write access: One capability that is currently mostly missing from all Semantic Web standards is the authorisation for read or write access to resources and RDF stores. Our empirical analysis shows, that if such capabilities are required for an application, then they will be implemented indirectly via other means. This will be addressed by future standards: FOAF+SSL [27] provides a generic, decentralised mechanism for authentication through the use of Semantic Web technologies. This is complemented by RDF Push [28], which describes how to authenticate updates to RDF stores.

E. Future approaches for applying the conceptual architecture

We propose software engineering and design approaches to facilitate the implementation of Semantic Web technologies in the future, and mitigate the identified implementation challenges. These approaches are: (1) guidelines, best practices and design patterns; (2) software libraries; and (3) software factories.

All of these approaches can be used to provide ready-made solutions and lower the entry barriers for software that is consuming from or publishing to the Web of Data. In addition, the reuse of such existing solutions can enable greater interoperability between Semantic Web applications, thus leading to a more coherent Web of Data.

1) More guidelines, best practices and design patterns: The Linked Data community has already developed some guidelines and collections of best practices for implementing Semantic Web technologies. These include publishing of Linked Data [26] and the naming of resources [29], and the Linked Data principles themselves. In the future this can be complemented by implementation-oriented guidelines such as patterns. A design pattern is a description of a solution for a reoccurring implementation problem [30]. A first collection of implementation-oriented patterns is provided by the Linked Data patterns collection[10], while [31] provides guidelines for requirements engineering and design of Semantic Web applications.

2) Software libraries: In order to go beyond libraries for accessing and persisting of RDF data, more software libraries in the future will directly provide reusable implementations of guidelines and patterns. Several such libraries are currently being developed such as ActiveRDF, which maps the native data model of a programming language to RDF [24], and the SIOC PHP API[11] for implementing access to SIOC data. Best practices for converting relational databases to RDF are implemented in the D2R server[12], while any23[13] implements best practices for conversion of many structured data formats to RDF. Various guidelines for accessing RDF are implemented by the Semantic Web Client Library[14], while the Silk framework[15] can be used for interlinking data sets. Additional projects for encoding best practices can be found on the LOD community wiki[16].

3) Software factories: Software factories provide the means to create complete and fully functioning applications by building on patterns, best practices and libraries [32]. They allow the assembly of complete applications from existing components and libraries that implement community best practices and patterns. Customisation for a domain or a specific application is possible through pre-defined extension points and hooks. While the majority of analysed applications uses RDF libraries, some components (e.g. navigation interface, homogenisation and data discovery services) are usually custom made for each application. A software factory for Semantic Web applications could provide a pre-made solution for each of the components from our conceptual architecture. Then the application developer could add the application and domain-specific logic and customise each of the components. This would allow rapid assembly of Semantic Web applications.

Similar benefits are already provided by modern Web application frameworks such as Ruby on Rails, PHPCake and Django. [33] presented a first prototype of such a Semantic Web software factory. A more current effort to create a software factory for the Web of Data is represented by Trice[17]. Trice uses the ARC RDF toolkit and SPARQL API for handling and persisting RDF data, and allows creating and customising of the user interface from within the framework. A different approach for rapidly creating complete applications is presented by content management systems (CMS) that include support for RDF and RDFa. One such CMS is Drupal 7[18],

[9] http://www.w3.org/TR/sparql11-query/

[10] http://patterns.dataincubator.org

[11] http://sioc-project.org/phpapi

[12] http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/

[13] http://any23.org/

[14] http://www4.wiwiss.fu-berlin.de/bizer/ng4j/semwebclient/

[15] http://www4.wiwiss.fu-berlin.de/bizer/silk/

[16] http://preview.tinyurl.com/LOD-wiki

[17] http://trice.semsol.org/

[18] http://preview.tinyurl.com/rdfa-in-drupal7

which follows existing community consensus for marking-up standard content types with RDFa.

V. Related work

We now discuss related work in the areas of empirical surveys about the adoption of Semantic Web technologies and architectures for Semantic Web applications.

1) Empirical surveys: Cardoso [34] presents the results of a survey of 627 Semantic Web researchers and practitioners carried out in January 2007. The questions from the survey covered the categories of demographics, tools, languages and ontologies. The goal of the survey was to characterise the uptake of Semantic Web technologies and the types of use-cases for which they were being deployed. Concrete applications were not taken into account. As the survey was carried out over a two months period of time, it does not allow conclusions about long term trends before or after the survey was taken.

Another similar survey of 257 participants (161 researchers and 96 industry-oriented participants) was published online[19] in 2009. The data for this survey was taken from a 3 month period. This survey had the goal of measuring the general awareness of Semantic Web technologies and social software in academia and enterprise. As such, the questions did not go into any implementation specific details.

Cunha [35] performed a survey of 35 applications from the "Semantic Web challenges" in 2003, 2004 and 2005. The goal of the survey was to cluster the applications based on their functionality and their architecture. The result is a list of 25 application categories with the number of applications per category for each year. This survey covers only 3 years, and it is not concerned with the adoption of Semantic Web technologies and capabilities, whereas our empirical survey provides empirical data on the uptake of e.g. data integration or the Linked Data principles.

2) Architectures for Semantic Web applications: García-Castro et. al [36] propose a component-based framework for developing Semantic Web applications. The framework consists of 32 components aligned on 7 dimensions, and is evaluated in 8 use-cases. While existing software tools and libraries that implement these components are identified, the identification of the components is not based on an empirical grounding.

Tran et. al [37] propose a layered architecture for ontology-based applications that is structured in a presentation layer, logic layer and data layer. It is based on best practices for service-oriented architecture and on the authors model of life-cycle management of ontologies, and evaluated in a use-case. However there is no empirical grounding for the architecture.

A similar approach is taken by Mika and Akkermans [38] who propose a layered architecture for ontology-based applications, with layers for the application, the middle-ware and the ontology. The architecture is based on a requirements analysis of KM applications.

Cunha [39] builds on his earlier survey in [35] and presents an UML architecture based on the 35 analysed applications. The goal of the architecture is to provide the foundation for a potential software framework, but no evaluation or implementation of the architecture is provided.

[19]http://preview.tinyurl.com/semweb-company-austria-survey

VI. Conclusions and future work

In this article we focused on the software architecture and engineering issues involved in designing and deploying Semantic Web applications. We argue that software engineering can unlock the full potential of the emerging Web of Data through more guidelines, best practices and patterns, and through software libraries and software factories that implement those guidelines and patterns.

Greater attention needs to be given to standardising the methods and components required to deploy Semantic technologies in order to prevent an adoption bottleneck. Modern Web application frameworks such as Ruby on Rails for Ruby, PHPCake for PHP and Django for Python provide standard components and architectural design patterns for rolling out database-driven Web applications. The equivalent frameworks for Semantic Web applications are still in their infancy.

As a guide to developers of Semantic applications, we present an architectural analysis of 124 Semantic applications. From this we put forward a conceptual architecture for Semantic Web applications, which is intended as a template of the typical high level components required for Semantic Web applications. Prospective developers can use it as a guideline when designing an application for the Web of Data. For experienced practitioners it provides a common terminology for decomposing and analysing the architecture of a Semantic Web application.

As future work, we will expand our empirical survey to include industrial prototypes and production systems. This will provide empirical data about the different priorities in adopting Semantic Web technologies between academia and industry. In addition, we plan to empirically evaluate refinements to our conceptual architecture, if required by newly emerging application scenarios. Finally, we will investigate the potential of providing a software factory to rapidly instantiate our architecture into customisable applications.

References

[1] T. Berners-Lee, J. A. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[2] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks, "The Semantic Web: The roles of XML and RDF," *IEEE Internet computing*, pp. 63–73, 2000.

[3] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data-The Story so far," *Journal on Semantic Web and Information Systems (in press)*, 2009.

[4] T. Berners-Lee, "Linked Data - Design Issues," http://www.w3.org/DesignIssues/LinkedData.html, 2006.

[5] C. Bizer, "The Emerging Web of Linked Data," *IEEE Intelligent Systems*, pp. 87–92, 2009.

[6] J. Hendler, "Web 3.0 Emerging," *IEEE Computer*, vol. 42, no. 1, pp. 111–113, 2009.

[7] D. I. K. Sjoberg, T. Dyba, and M. Jorgensen, "The Future of Empirical Methods in Software Engineering Research," *Future of Software Engineering*, pp. 358–378, 2007.

[8] M. Klein and U. Visser, "Guest Editors' Introduction: Semantic Web Challenge 2003," *IEEE Intelligent Systems*, pp. 31–33, 2004.

[9] D. Soni, R. L. Nord, and C. Hofmeister, "Software architecture in industrial applications," *International Conference on Software Engineering*, 1995.

[10] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, UC Irvine, 2000.

[11] C. Bizer and D. Westphal, "Developers Guide to Semantic Web Toolkits for different Programming Languages," Freie Universität Berlin, Tech. Rep., 2007.
[12] J. Broekstra, A. Kampman, and F. van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," in *International Semantic Web Conference*, 2002.
[13] K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds, "Efficient RDF storage and retrieval in Jena2," in *Workshop on Semantic Web and Databases*, 2003.
[14] A. Doan and A. Y. Halevy, "Semantic integration research in the database community: A brief survey," *AI Magazine*, pp. 83–94, 2005.
[15] "CROSI - Capturing, Representing and Operationalising Semantic Integration, The University of Southampton and Hewlett Packard Laboratories," http://eprints.ecs.soton.ac.uk/10842/1/crosi-survey.pdf, 2005.
[16] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K.-U. Sattler, and J. Umbrich, "Data Summaries for On-demand Queries over Linked Data," in *World Wide Web Conference*, 2010.
[17] J. Umbrich, M. Hausenblas, A. Hogan, A. Polleres, and S. Decker, "Towards Dataset Dynamics: Change Frequency of Linked Open Data Sources," in *Workshop on Linked Data on the Web*, 2010.
[18] C. Mangold, "A survey and classification of semantic search approaches," *International Journal of Metadata, Semantics and Ontologies*, vol. 2, no. 1, pp. 23–34, 2007.
[19] M. Hildebrand, J. van Ossenbruggen, and L. Hardman, "An analysis of search-based user interaction on the Semantic Web," CWI Information Systems, Tech. Rep., 2007.
[20] M. Hildebrand and J. van Ossenbruggen, "Configuring Semantic Web interfaces by data mapping," in *Workshop on Visual Interfaces to the Social and the Semantic Web*, 2009.
[21] D. Huynh, D. Karger, and R. Miller, "Exhibit: lightweight structured data publishing," in *World Wide Web Conference*, 2007.
[22] A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure, "SEemantic portAL-the SEAL approach," *Spinning The Semantic Web: Bringing The World Wide Web To Its Full Potential*, 2005.
[23] M. Krötzsch, D. Vrandečić, and M. Völkel, "Semantic MediaWiki," in *International Semantic Web Conference*, 2006.
[24] E. Oren, B. Heitmann, and S. Decker, "ActiveRDF: embedding Semantic Web data into object-oriented languages," *Journal of Web Semantics*, 2008.
[25] A. Leff and J. Rayfield, "Web-application development using the Model/View/Controller design pattern," *International Enterprise Distributed Object Computing Conference*, pp. 118–127, 2001.
[26] C. Bizer, R. Cyganiak, and T. Heath, "How to Publish Linked Data on the Web," FU Berlin, Tech. Rep., 2007.
[27] H. Story, B. Harbulot, I. Jacobi, and M. Jones, "FOAF+SSL: RESTful Authentication for the Social Web," in *Workshop on Trust and Privacy on the Social and Semantic Web*, 2009.
[28] O.-E. Ureche, A. Iqbal, R. Cyganiak, and M. Hausenblas, "Accessing Site-Specific APIs Through Write-Wrappers From The Web of Data," in *Workshop on Semantics for the Rest of Us*, 2009.
[29] L. Sauermann, R. Cyganiak, and M. Volkel, "Cool URIs for the Semantic Web," W3C, W3C Note, 2008.
[30] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1995.
[31] Ó. Muñoz-García and R. Garcia-Castro, "Guidelines for the Specification and Design of Large-Scale Semantic Applications," in *Asian Semantic Web Conference*, 2009.
[32] J. Greenfield and K. Short, "Software factories: assembling applications with patterns, models, frameworks and tools," in *Conference on Object-oriented programming, systems, languages, and applications*, 2003.
[33] E. Oren, A. Haller, M. Hauswirth, B. Heitmann, S. Decker, and C. Mesnage, "A Flexible Integration Framework for Semantic Web 2.0 Applications," *IEEE Software*, 2007.
[34] J. Cardoso, "The Semantic Web Vision: Where Are We?" *IEEE Intelligent Systems*, vol. 22, pp. 84–88, 2007.
[35] L. M. Cunha and C. J. P. de Lucena, "Cluster The Semantic Web Challenges Applications: Architecture and Metadata Overview," Pontificia Universidade Catolica do Rio de Janeiro, Tech. Rep., 2006.
[36] R. Garcia-Castro, A. Gómez-Pérez, Ó. Muñoz-García, and L. J. B. Nixon, "Towards a Component-Based Framework for Developing Semantic Web Applications," in *Asian Semantic Web Conference*, 2008.
[37] T. Tran, P. Haase, H. Lewen, Ó. Muñoz-García, A. Gómez-Pérez, and R. Studer, "Lifecycle-Support in Architectures for Ontology-Based Information Systems," in *International Semantic Web Conference*, 2007.
[38] P. Mika and H. Akkermans, "D1.2 Analysis of the State-of-the-Art in Ontology-based Knowledge Management," SWAP Project, Tech. Rep., February 2003.
[39] L. M. Cunha and C. J. P. de Lucena, "A Semantic Web Application Framework," Pontificia Universidade Catolica do Rio de Janeiro, Tech. Rep., 2007.

Benjamin Heitmann received the M.Sc. degree in informatics from the University of Karlsruhe, Karlsruhe, Germany, in 2008.

He is currently working toward the Ph.D. degree in the Digital Enterprise Research Institute (DERI), NUI Galway, Galway, Ireland. His current research interests include software architecture, adaptive personalisation and the evolution of the Web of Data.

Richard Cyganiak is a Linked Data Technologist at the Digital Enterprise Research Institute (DERI), NUI Galway, Ireland.

He is a co-founder of the Linking Open Data initiative, a fellow of the Web Science Research Initiative, and founder or contributor to several Semantic Web related software projects D2RQ, Pubby, Neologism, Sigma, and Sindice. As a researcher, his interests are in web-scale data integration and in the theory of web architecture.

Conor Hayes received the M.Sc. and the Ph.D. degree in intelligent web systems from Trinity College Dublin, Dublin, Ireland, in 1997, and 2004, respectively.

He is currently a research fellow, adjunct lecturer and leader of the Information Mining and Retrieval Unit at the Digital Enterprise Research Institute (DERI), NUI Galway, Galway, Ireland. His current research interests include tracking and modelling of information diffusion, behavioural analysis and modelling in social networks, analysis of cross-community dynamics and case-based reasoning methodologies for the Semantic Web.

Stefan Decker received the M.Sc. in computer science from the University of Kaiserslautern, Kaiserslautern, Germany, and the Ph.D. degree in computer science from the University of Karlsruhe, Karlsruhe, Germany, in 1995, and 2002, respectively.

He is a full professor at NUI Galway, Galway, Ireland, and the director of the Digital Enterprise Research Institute (DERI), NUI Galway, Galway, Ireland. Previously he worked at ISI, University of Southern California (2 years, Research Assistant Professor and Computer Scientist), Stanford University, Computer Science Department (Database Group) (3 Years, PostDoc and Research Associate), and Institute AIFB, University of Karlsruhe (4 years, PhD Student and Junior Researcher). He is one of the most widely cited Semantic Web scientists, and his current research interests include semantics in collaborative systems, Web 2.0, and distributed systems.