IKS Developing Semantic CMS

APPLICATIONS

The IKS Handbook

2013

Disclaimer

Developing Semantic CMS Applications The IKS Handbook

EDITORS:
WERNHER BEHRENDT
VIOLETA DAMJANOVIC

Imprint

This report is a product of the IKS project that is supported and partly funded by the EU IST Seventh Framework Programme of the European Commission

Copyright
Salzburg Research Forschungsgesellschaft m.b.H on behalf of the IKS Consortium.
Publisher Salzburg Research Forschungsgesellschaft m.b.H

March 2013

ISBN: 978-3-902448-35-4

Design and Layout
John Pereira, Salzburg Research Forschungsgesellschaft m.b.H

Copyright for the images
All images are copyright IKS Consortium unless otherwise specified in text.

Editors
Wernher Behrendt and Violeta Damjanovic of Salzburg Research Forschungsgesellschaft m.b.H.

Foreword

This book is intended for developers and CTOs who need to deal with Content Management Systems (CMS) and with any kind of "smart" applications that combine web-based information sources with some Information System that is being built or adapted, for their own organization or for a customer.

The book summarizes the results of four years of collaborative technology development between CMS providers and research organizations in Europe, plus the input from 40 early adopter organizations, worldwide.

The book does not give you academic depth: we have opted for enough detail for readers to understand the system and its components, and we have tried to satisfy the practitioner's interest in IKS technology. The book has six sections:

- Initial concepts - to acquaint you with different notions of the term "semantics";
- Knowledge Representation and examples of "Semantic Web" applications;
- Building Semantic Components and making them usable in a customer CMS
- Using Apache Stanbol and VIE as semantic components for real-world use;
- Showcases of IKS semantic technologies in use;
- IKS, Semantic Web, Linked Data, Artificial Intelligence – A critical appraisal.

The book is based on several pieces of open source software, in particular: on Apache Stanbol, a set of "semantic engines" that help software developers to lift textual information to structured, computable representations; on the VIE libraries for connecting HTML5 based web interfaces with semantic engines, be they from Stanbol or from elsewhere; on other open source software such as Apache Tika, Apache Chemistry or jQuery

The IKS software is available under permissive licensing on Apache and on github.

In a Nutshelll

"Interactive Knowledge Stack" (**IKS**) was a four-year research project targeting small to medium CMS providers in Europe with the aim of providing technology platforms for content and knowledge management to thousands of end user organisations. The starting point was that CMS technology platforms often lack the capability for semantic web enabled, intelligent content, and therefore lack the capacity for users to interact with the content at the user's knowledge level.

The objective of IKS therefore, was to bring semantic capabilities to current CMS frameworks. IKS postulated a "Semantic CMS Technology Stack" which merges the advances in semantic web infrastructure and services with CMS industry needs of coherent architectures that fitted into existing technology landscapes. At the end of the four years, IKS has not only provided specifications, but also modular instantiations of the **IKS Stack**. Prototype solutions for industrial use cases were developed, ranging from semantics-based web site management to smart, online holiday booking systems and to demonstrators for future, ambient intelligence infotainment in the home.

The project's biggest success is the launching of the **Apache Stanbol** top-level project, which graduated after incubation in late 2010, to full Apache project status in September 2012. Similarly, the development of the "Vienna IKS Editables" (**VIE**) has given the CMS community an easy path from HTML5-based User Interfaces to semantic back-ends, a development led by a Finnish small-to-medium enterprise. The VIE libraries have found their way into top-ranking CMSs such as Drupal and Typo3. The ground breaking approach of involving 40 further CMS providers and end user organizations for validation of the technology has helped to disseminate the tangible results of IKS quickly, amongst a large community.

On the following pages, we try to give readers a fast way into IKS, so that they can also benefit from the results of 6.5 m € of European Funding which accounted for approximately 75% of the IKS Budget.

Table of Contents Semantic

CHAPTER 1: Initial Concepts.. 8

1.1 Semantics in Linguistics, Computer Science and in Web Engineering 9

1.2 Semantic Web: from Tim Berners Lee's Vision to Today's State of the Art 9

1.3 Linked Data 13

CHAPTER 2: Knowledge Representation Methods and Techniques 15

2.1 Semantics and Content Management Systems 15

2.2 Methodologies for Knowledge Modeling 16

2.3 Semantic Technologies and Tools 20

2.4 Semantic Web Success Stories 26

CHAPTER 3: IKS Methodology for Building Semantic Components into CMS 33

3.1 High Level Requirements 34

3.2 IKS Alpha: Refactoring CMS and Semantic Web Technology 38

3.3 The IKS Reference Architecture 42

3.4 The IKS Reference Implementation 46

CHAPTER 4: Apache Stanbol and VIE in a Semantic CMS Technology Stack 50

4.1 Foundational Components of IKS 50

4.2 Ambient Intelligence Components of IKS 65

CHAPTER 5: Showcases 71

5.1 Showcasing Horizontal Applications of IKS 71

5.2 Showcasing Vertical Applications of IKS 78

CHAPTER 6: Beyond CMS with semantic extensions 86

6.1 The Roaring Success of Semantic Web Technologies 86

6.2 The Dismal Failure of Semantic Web Research 87

6.3 A Pragmatist's View on Semantic Technologies in Web Content Management 98

6.4 What's Next? 106

References 109

CHAPTER 1: Initial Concepts

This book is about developing semantic CMSs and their applications. Semantic CMSs differ from traditional CMSs by their capability to interact with the content, as well as to automatically extract, manage, and store semantic metadata about content [IKS-D4.2]. While traditional CMSs provide management tools for document types and workflows, semantic CMSs promise knowledge management at the level of real-world entities, through the use of ontologies. So, while traditional CMSs target the “document” as an atomic unit, semantic CMS treat real-world entities as atomic units. Actually, documents also become real-world entities that are “about” other real-world entities!

Semantics comes to CMS via Semantic Web technologies. The overall idea is to utilize ontologies that provide vocabularies, definitions, and constraints describing the domain of interest. Information resources, agents, and web-based applications can commit to these ontologies in order to reuse data and knowledge effectively [HEHU03]. Semantic Web technologies embrace a distributed approach to creating standard vocabularies that, when integrated within CMS, can provide formal relationships to enrich the content.

A semantic CMS is designed to manage two types of data [IKS-D5.0]. The first data type is content by itself, i.e. the kind of data that is traditionally managed by a CMS. Content can be text or any other kind of binary data like images, video, or sound. CMSs implement content (business) lifecycles, manage editing responsibilities, access controls, and output channels. The second kind of data (and this is typically missing in a CMS, or very rudimentarily managed) is knowledge about the content that is stored within the CMS. A semantic CMS manages such knowledge explicitly and offers features to gain further knowledge from the available content.

In this book, we describe our development experience in building semantic components and integrating them into existing CMSs through RESTful Web services. We discuss the IKS Reference Architecture and its Reference Implementation for managing and deploying semantic components and vocabularies. In addition, we present the results of the two major sub-projects of IKS: the semantic back-end of Apache Stanbol and the semantically enabled front-end technology, called VIE (Vienna IKS Editables). Finally, we present several vertical and horizontal demonstrators of the IKS-based semantic CMS applications.

1.1 Semantics in Linguistics, Computer Science and in Web Engineering

The term “semantics” (as in Semantic Web) is not an invention of web engineers. In linguistics, semantics is a research field about how meaning is constructed in the human mind, from words and sentences. In computing, “formal semantics” is the study how computers behave when executing programs written in some formal language. One of the fundamental differences between natural languages and programming languages is that in computing, we make the *compositionality assumption* according to which each sentence (which is equal to a program statement) is composed in a precise fashion, from the meanings of the programming keywords. Thus, meaning in computing can be formally constructed. Such a strong claim is difficult to maintain for natural language where the meaning of words and phrases is known to change over time, and where competing meanings (ambiguities) are often resolved through applying other knowledge (“context”).

The notion of Semantic Web attempts to bring formal knowledge representation and natural language understanding closer to each other, in order to make it possible to represent and encode everyday knowledge in a way that is process-able and computable by machines.

1.2 Semantic Web: from Tim Berners-Lee’s Vision to Today’s State of the Art

In 2001, an influential article in Scientific American, co-authored by Tim Berners-Lee [BERN01], marked the start of a 10+ year period of research into extending the WWW infrastructure so as to making it more “intelligent”, i.e. enabling software robots to automatically navigate and “work their way” through the WWW, doing useful things for their owners. The suggestive cover of that issue of Scientific American showed a computer screen with the words: *“I know what you mean …”* and the cover headline was *“Get the Idea? (Tomorrow’s web will)”*. While we encourage readers to revisit the original article, we warn you that what is now available as “semantic technologies” and “Linked Open Data” has only one common denominator: the use of the RDF notation for expressing things that have specific meaning. What makes the current Semantic Web work is a mix of natural language processing capabilities, the use of controlled Web vocabularies (ontologies) and an increasing number of data sets that have encoded partial semantics - usually just enough to build applications that “mash up” those partial semantics in order to create added value for a specific use.

Web ontologies have become the most widely accepted standard artifacts for representing and reasoning upon knowledge, which is formally rendered as a set of concepts defined either within a given domain or across multiple domains, along with the logical relationships between and constraints upon, these concepts. A widely accepted definition of ontology from the literature is the one given by Tom Gruber [GRU93]: “*An ontology is an explicit specification of a conceptualization.*” More recently, Tom Gruber has elaborated on that definition and in [GRU09] writes: “*In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members).”*

The core feature of ontologies is the possibility to define formal models of knowledge domains by combining any type and amount of semantic structures (such as taxonomies, mereonomies, non-hierarchical relations such as equals and opposites) on the basis of relationships between “things”. All such “things” are then referenced by identifiers complying to a uniform mechanism, i.e., Uniform Resource Identifiers (URI).

Each ontology provides the vocabulary (or labels) for referring to the terms in a subject area, as well as the logical statements that describe what the terms are, how they are related to each other, etc. One of the central roles of ontologies is to establish further levels of interoperability, i.e. semantic interoperability, between agents and applications on the emerging Semantic Web [BERN01], as well as to add a further representation and inference layers on top of the Web's current layers [DECK00], [HEND01].

Social Semantic Web. Since about 2005, web researchers and developers have started to combine Social Web (or Web 2.0) and Semantic Web principles, techniques, approaches and tools into the so-called Social Semantic Web. The Social Web is a platform for social and collaborative exchange [OREI05] where users meet, collaborate, interact and most importantly create content and share knowledge through, e.g., wikis, blogs, photo- and video sharing services [OP4L-D1.1]. The Social Web transforms the "old" model of the Web into a platform for social and collaborative exchange. Popular social websites, such as Facebook, Flickr and YouTube, enable people to keep in touch with friends and share content. Other services such as blogs, wikis, video and photo sharing that together enable what recently has been defined as "life-streaming" - allowing novice users to easily create, publish and share their own content. Furthermore, users are able to easily annotate and share Web resources using social bookmarking and tagging, thus creating metadata for Web content commonly referred to as "folksonomies". However, Social Web technologies in general, and collaborative tagging in particular, suffer from the problems of ambiguity of meanings. For instance collaborative tags are often ambiguous due to their lack of precisely defined semantics. Moreover, they lack a coherent categorization scheme, and they require significant time and a sizeable community to be used effectively [STJO09] [OP4L-D1.1].

Among the key representatives of the Social Web are mash-ups - Web applications allowing users to combine and integrate different types of data, often originating from different sources. Map-based mash-ups, in which maps are overlaid with other information, are one emerging type of tools. Tools, such as Google Refine[1], or Yahoo Pipes[2] allow individuals to aggregate data, find new meanings or interpretations, and present the data in interesting ways. The suite of tools developed in the scope of MIT's SIMILE[3] project (such as Exhibit [HUYN07a] and Potluck [HUYN07b]) facilitates the creation of Semantic Web mash-ups. By leveraging Semantic Web technologies (primarily RDF and SPARQL), these mash-ups are more dynamic and flexible than those offered by simpler Web 2.0 tools and services. For example, the Potluck tool lets casual end-users (i.e. nonprogrammers) easily make mash-ups of structured, semantically rich data, often expressed in RDF or JSON[4] format. Potluck acknowledges the fact that the real-world RDF is messy, "broken perhaps not just in syntax but also in semantics" [HUYN07b], and empowers users to deal with this problem by providing them with visual editing facilities. In particular, the tool assumes an iterative process of data integration in which the user can take advantage of the tool's rich visualization capabilities to explore the data, identify data of interest as well as merge, align and/or clean up the data in an easy and intuitive manner.

Today, many projects deal with topics related to the Social Semantic Web. For instance, DBpedia[5] is a large-scale semantic knowledge base, which re-structures knowledge that has been socially created on Wikipedia. DBpedia takes advantage of the common patterns and templates used by Wikipedia authors, to gather the semi-structured information into a formal knowledge base. The result is a huge database of shared knowledge, which allows "intelligent" queries such as: "List the 19th century poets from England" [AUER06]. With its capability to answer very specific queries, DBpedia can serve as a learning tool

[1] Google Refine wepage: http://code.google.com/p/google-refine/

[2] Yahoo Pipes webpage: http://pipes.yahoo.com/pipes/

[3] SIMILE project webpage: http://simile.mit.edu/

[4] JSON webpage http://json.org

[5] DBpedia webpage: http://dbpedia.org

and is an excellent example of the advantages that the Social Semantic Web paradigm brings to various domains.

1.3 Linked Data

As explained in [HEBI09]: "*Linked Data provides a publishing paradigm in which not only documents, but also data, can be a first class citizen of the Web, thereby enabling the extension of the Web with a global data space based on open standards - the Web of Data.*" The term Linked Data refers to a set of best practices for publishing and interlinking structured data on the Web that were introduced by Tim Berners-Lee and have become known as the *Linked Data principles* (rules) [LEE06]. These principles provide guidelines on how to use standardized Web technologies to set data-level links between data from different sources. Due to the fact that data from different sources is connected by links, it is possible to crawl the data space, fuse data about entities from different sources, and provide expressive query capabilities over aggregated data, similarly to how a local database is queried today. Linked Data applications discover new data sources at runtime by following data-level links, and can thus deliver more complete answers as new data sources appear on the Web [BIZE09].

The publication of Linked Data is loosely coordinated by the World Wide Web Consortium's (W3C) Linking Open Data (LOD) project. Its goal is to bootstrap the Web of Linked Data by identifying existing data sets that are available under open licenses, converting them to RDF according to the Linked Data principles, and publishing them on the Web. Major publishers and consumers of Linked Data today are classified according to their field of interests (see Table 1 in the Appendix for more details):

- British Broadcasting Corporation (BBC, Media Metadata): The BBC Programmes and Music sites provide data about episodes of radio and TV programs. The data is interlinked with MusicBrainz, an open-license music database, and DBpedia, a Linked Data version of Wikipedia. The links between

BBC Music, MusicBrainz, and DBpedia let applications retrieve and combine data about artists from all three sources;

- **The US Library of Congress** and **the German National Library of Economics** have published their subject heading taxonomies as Linked Data;

- **Amazon and Google Base APIs**: The RDF Book Mashup, a wrapper around the Amazon and Google Base APIs, provides Linked Data about books;

- **The Open Archives Initiative - Object Reuse and Exchange standard (OAI-ORE)** is also based on the Linked Data principles;

- **W3C Linking Open Drug Data**: Within the W3C Linking Open Drug Data effort, the pharmaceutical companies Eli Lilly, AstraZeneca, and Johnson & Johnson cooperate to interlink open- license data about drugs and clinical trials to ease drug discovery.

From these examples it can be seen that semantic CMS applications will soon be the norm rather than the exception. In other words, future CMS will be expected to deal with data and text that have varying degrees of semantic structure. IKS also contributes methods and tools for this.

CHAPTER 2: Knowledge Representation Methods and Techniques

The major challenge of the IKS project was to hide the complexity of ontology engineering and Semantic Web technologies from the developers of semantic CMSs. As market, modularity, reusability and application interoperability are critically important aspects for small to medium enterprises (SMEs) around CMSs, this became important driver of the whole project.

This chapter briefly surveys Semantic Web technologies, their techniques, formalisms, standards and applications targeting knowledge representation (KR) technologists around CMSs.

2.1 Semantics and Content Management Systems

A CMS is a collection of software applications and their functionalities that assists end-users in creating, editing, publishing and managing content within a collaborative environment [IKS-D3.2]. Such systems can be classified with respect to the business needs and production environments. For example, we distinguish between:

- Web Content Management Systems (WCMS) for publishing content on websites;
- Mobile Content Management Systems (MCMS) that deliver content to mobile devices, i.e. smartphones and PDAs;
- Document Management Systems (DMS), with a focus on the storage and management of electronic documents, either authored in electronic form or ported from paper documents;
- Enterprise Content Management Systems (ECMS) for dealing with content that is related to the organizational processes of an enterprise.

The above classification of CMSs does not necessarily constrain the business domain in which CMS can be used. For example, content of e-commerce portals, libraries and news agencies can be managed by a CMS of any kind, whereas media companies would gain a lesser benefit from DMSs. However, knowledge that is present in CMSs needs to be formally modeled in order to provide features such as reasoning and learning in CMSs. Hence, we can postulate that most, if not all of the above CMS categories, are grounded on shared knowledge management schemes, or knowledge patterns. Such schemes describe the usage context, language constructs, authored data, the users of a system, as well as the system itself. In addition, such knowledge schemes are subject to representation by specific knowledge modeling methodologies, technologies and formats.

2.2 Methodologies for Knowledge Modeling

Rapid changes, evolution, diversity, entropy of systems create many difficulties for people to recognize, understand and model their knowledge domains. Therefore, knowledge modeling and engineering bridges many complex domains and implicit knowledge existing in these domains, providing formal validation of knowledge models in terms of their logical correctness [SOWA06].

CMSs are not specifically tailored around the roles and capabilities of knowledge engineering. Hence, high interaction between business domain experts and knowledge engineers is required in all tasks that relate to any form of knowledge modeling. Therefore, we briefly survey several knowledge engineering methodologies supporting semantic enhancement of CMSs [IKS-D3.2].

On-To-Knowledge [DFV02] [SS02] built an ontology-based tool environment to improve knowledge management, dealing with large numbers of heterogeneous, distributed, and semi-structured documents, which are typically found in large company intranets and on the Web. The On-To-Knowledge project aimed to provide (i) a toolset for semantic information processing and user access, (ii) OIL, an ontology-based inference layer for the Web, and (iii) an associated methodology and validation by industrial case studies.

METHONTOLOGY [FGPJ97] is a methodology enabling the construction of ontologies at the knowledge level. METHONTOLOGY identifies the set of activities to be carried out, based on the main activities identified by the software development process and used in knowledge engineering methodologies.

A similar approach was taken in **Grüninger's and Fox's Methodology** [GRÜN94]. This methodology introduced the use of competency questions for defining atomic problems to be directly solved by ontologies. Essentially, it involved building a logical model of the knowledge that needs to be specified by means of the ontology.

The Methodology proposed **by Uschold and King** is based on the experience in developing the Enterprise ontology [USC95].

Knowledge modeling methodology that follows the **SENSUS approach** is based on SENSUS ontology, which is an ontology aimed to be used in Natural Language Processing (NLP). SENSUS was developed at the ISI (Information Sciences Institute), providing a broad-based conceptual structure for developing machine translators [KNI94] [KNI95].

The Methodology of Amaya Berneras et al., was developed within the Esprit KACTUS project [KAC96]. One of the objectives of the KACTUS project was to investigate the feasibility of knowledge reuse in complex technical systems, as well as the role of ontologies to support it [SCH95]. Such an approach to developing ontologies is conditioned by application development [FLGP02]. Every time an application is developed, the following steps must be taken:

Specification of the application, which provides an application context and a view of the components that are modeled by the application;

Preliminary application design based on relevant top-level ontological categories;

Ontology refinement and structuring in order to arrive at a definitive design;

The principles of minimum coupling should be used to assure that the modules are not dependent on each other and are as coherent as possible.

The Diligent Methodology [PTSS04] has its focus on the evolution of ontologies and identified those arguments that need to be exchanged during the evolution of ontologies, i.e. arguments supporting the discussion of ontology changes.

Following up on the above-mentioned ontology engineering methodologies, the FP7 EU NeOn project[6] introduced the **NEON Methodology** for collaborative ontology networks and semantic applications. The aim was to assure that collaborative methodologies are usable for ontology engineers, as well as for software practitioners. The NEON methodology [SFGP09] supports collaborative aspects of ontology development and reuse, as well as the dynamic evolution of ontology networks in distributed environments via contextual information. The methodology specifically addresses the development process and different life cycle models, methods, techniques and tools that can be used while building ontology networks. The NEON methodology makes extensive use of the Ontology Design Patterns (ODPs) [GP09] [IKS-D3.2]. Most pattern-based methods in ontology engineering cover primarily the logical level by providing support for ontology learning, enrichment, etc. [NRB09] [BLO09], while putting little or no focus on solving concrete modeling problems, i.e. the content level. Pattern support provided by these methodologies is automatic for the most part, such as the usage of lexico-syntactic patterns to identify concepts or relations between concepts in a natural language text [CIM06]. In 1997, Clark [CP97] proposed a method for constructing ontologies based on patterns, although these were assumed to be non-evolving sets, mostly defined with a top-down approach. Other examples include the Ontology Pre-Processor Language (OPPL) [IRS09] and methods for applying it as a means for logical ODP reuse, as well as the proposal for a high-level pattern language by Noppens and Liebig [LVHN05]. Use of ODPs has also been spotted in some ontology engineering environments, such as the logical pattern templates in Protégé3[7], and the template wizard supporting OPPL pattern definitions in Protégé4.

The eXtreme Design (XD) Methodology [PDGB09], developed also in the context of the NeOn project, is a pattern-based design methodology that uses a set of competency questions as a reference source for requirement analysis. This methodology focuses on producing modular networked ontologies that extensively reuse ODPs. XD relies on the application of design best practices and a test-driven

[6] FP7 EU NeOn project: http://www.neon-project.org

[7] Protege website: http://protege.stanford.edu/

development approach for ontologies. Furthermore, the methodology takes the basic principles of the eXtreme Programming[8] methodology for software development. The XD methodology is supported by the XD Tools[9] available as an Eclipse plugin, which is compatible with the NeOn Toolkit. In addition to patterns for modeling, patterns for the usage of ontologies have been also proposed. For example, the reasoning patterns shown in [VHTTW09] describe primitive reasoning services, such as classification, realization, mapping, and their composition into more high-level reasoning patterns, such as search, browsing, personalization, integration, and recommendation. This can both be seen as a way of standardizing typical reasoning tasks, but also as a way to hide the underlying primitive steps, i.e., to hide some of the complexity from a user or system developer.

Conclusions on methodologies for Knowledge Modeling. According to the analysis of methodologies for knowledge modeling and ontology engineering presented in [FLGP02], the following conclusions can be drawn:

1. None of the existing methodologies are fully mature (e.g. compared with the *IEEE Standard for Developing Software Life Cycle Processes, 1074-1995*), although the following scale can be established:

   METHONTOLOGY is the most mature knowledge modeling methodology, recommendations for the pre-development processes are required, while some activities and techniques should be specified in more detail. METHONTOLOGY is recommended by the IEEE Foundation for Intelligent Physical Agents (FIPA) (c.f. http://www.fipa.org/).

   In Grüninger and Fox' s methodology [GRÜN94], neither the activities nor the techniques for performing such activities (for example, techniques for formulating the competency questions) are described in detail.

   Uschold and King's methodology [USC95] has the same omissions as the above methodology and is even less detailed.

[8] XP: http://www.extremeprogramming.org/

[9] XD Tools: http://stlab.istc.cnr.it/stlab/XDTools

Berneras et al.’s methodology [SCH95] [KAC96], apart from the above omissions, has not been used to build many ontologies and applications.

2. The proposals are not unified. At present each group applies its own methodology. This is exacerbated by the fact that none have reached maturity. Therefore, additional efforts are required along the lines of unifying the existing methodologies.

3. A preliminary attempt to unify two methodologies was described in [USC96]. Its disadvantage was that the new synthesized methodology was not an actual methodology; it was a conception of a potential methodology. This points that the best we can do is, perhaps, to have several widely accepted methodologies rather than one standardized.

Nonetheless, there is a starting point for solving the above problems. We have a series of methodologies that can be used as reference points for developing methodologies that are adaptable to different ontology types in different settings.

2.3 Semantic Technologies and Tools

Ontologies provide a number of useful features for knowledge-based intelligent systems, knowledge representation and engineering. In order to define the semantics of knowledge and content used in these systems, several languages were developed so far, such as Resource Description Framework (RDF), Web Ontology Language (OWL), and more.

RDF is the main formalism for rendering Web ontologies. It belongs to a family of World Wide Web Consortium (W3C) specifications for representing information on the Web. It offers a simple graph model, which consists of nodes (i.e. resources or literals) and binary relations (i.e. statements) (c.f. http://www.w3.org/RDF/). The RDF data model exploits a recurring linguistic paradigm in Western languages by representing all facts as *subject-predicate-object* expressions, called triples. In an RDF triple, subject and predicate represent resources identified by URIs, while the object can either identify a resource or a literal value. Since the object of one resource triple can be the subject of one or more other triples, the resulting triple set forms a graph in which resources represent nodes and arcs depend on their role in each triple. In that way, RDF represents a type of Semantic Network, similar to the *relational model of data* that is given in [CODD70]. Such a simple model embodies a small amount of built-in semantics and offers great freedom in creating customized extensions [DKDA05]. For example, John Sowa identifies six categories of Semantic Networks, which are based on relation semantics [SOWA02]:

> *Definitional networks* building taxonomies for conceptualisms with inheritance (subclass) and membership (instance) relations;
>
> *Assertional networks* representing cognitive assertions about the world with modal operators;
>
> *Implicational networks* focusing on implication relations, e.g. belief network;
>
> *Executable networks* focusing on temporal dependence relations, e.g. flowchart, PetriNet;
>
> *Learning networks* focusing on causal relations encoded in numerical value, e.g. neural network, and
>
> *Hybrid networks* combining features of the above types.

In the Semantic Web, most ontologies are defined using languages such as RDF Schema RDF(S) and/or OWL, thus falling in the category of the definitional networks. The category of assertional networks emerges in the context of sharing instance data and evaluating trustworthiness of such data (c.f. [YOVA02][JEBJ03][MARP03][SGUC03][CBHS04][DKFJ05]), while the third category of Semantic Networks (c.f. implicational networks) is focused on ontology mapping [ZHYU04][ZHYR04].

The **RDF Schema (RDFS)** provides a base mechanism for sharing terminologies and other relationships between resources (c.f. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/). RDFS provides specifications and constructs for expressing classes and subsumption relationships (thus being able to define taxonomies), and domain and range definitions for properties. RDFS is used to augment RDF to provide better support for definition and classification [LAMG01]. In addition to inheriting basic features from Frame Systems (c.f. [LAMG01]), RDFS provides ontology constructs that make relations less dependent on concepts. In other words, users can define relations as an instance of *rdf:Property,* describe inheritance relations between relations using *rdfs:subPropertyOf,* and associate defined relations with classes using *rdfs:domain* or *rdfs:range.* In that way, releasing bulks of semantically structured corporate and public knowledge as RDF datasets, is one aspect of the practice of data interoperability. Reuse of machine-readable data is best performed along with expressing their relationships with human-readable data, which requires formal methods for interlinking the two worlds [IKS-D3.2]. The major advancements of such formalisms for embedding knowledge have been observed in the form of markups for HTML and XML documents.

Microformats represent a class of markup techniques for embedding formalized knowledge into HTML pages [ALLS07]. A single microformat denotes a custom finite vocabulary whose terms are embedded within (X)HTML elements and encoded as attributes of some of their markup tags, namely @class and @rel. Microformats have gained a wide adoption over the last half-decade, thus leading to a centralized open community for the development and sharing of microformat vocabularies. Custom microformat vocabularies are developed for the semantic markup of specific types of information. The most widely used microformats are: *hCalendar* for events, *hProduct f*or details and features of products, *XOXO* for lists and blog rolls, and *geo* for geographical coordinates. At the same time, the proliferation of microformats is generally discouraged, as it may dramatically increase the likelihood of class collisions, which are generally harder to keep under control.

Partly inspired by microformats, **embedded RDF (eRDF)** has been the first real attempt at lifting the practice of embedding knowledge to a general-purpose level (c.f. http://research.talis.com/2005/erdf/wiki/Main/RdfInHtml). It was devised by Ian Davis in 2005 as a markup technique for embedding arbitrary RDF in (X)HTML documents.

RDFa [ABMP08] is an alternative to eRDF, which uses a specific XML attributes (hence the ‘a’) to convey RDF triples in XML and XHTML elements, with an adaptation to non-XML versions of HTML (c.f. http://www.w3.org/TR/2010/WD-rdfa-in-html-20100304/). Formally it is intended to improve upon eRDF by supporting typed literals and blank nodes. As opposed to eRDF, RDFa is under direct W3C support, and is being natively supported in open-source CMSs, such as Drupal. Search platforms like Yahoo! SearchMonkey [MIK09] are parsing and consuming RDFa for the delivery of structured search results. A MediaWiki extension for rendering Semantic MediaWiki markup as RDFa is also available (c.f. http://www.mediawiki.org/wiki/Extension:RDFa).

The **Hyper Text Markup Language version 5 (HTML5)** introduced more powerful multimedia support mechanisms directly at the HTML layer [IKS-D3.1]. Apart from multimedia, HTML5 supports semantic annotation. This is done by including RDFa in HTML5 documents as a way to represent RDF data as XHTML (and HTML5) (c.f. http://www.w3.org/TR/xhtml-rdfa-primer/). The magic of such specification is bringing in singular document information about presentation and meaning, so that browser capability for processing documents can be enhanced. Recently, W3C published the first draft for RDFa API (c.f. http://www.w3.org/TR/2010/WD-rdfa-api-20100608/), which provides automatic extraction and manipulation functionality. However, RDFa was not incorporated as the official semantic markup for HTML5, which is instead represented by microdata (c.f. http://dev.w3.org/html5/md).

Microdata essentially defines how HTML5 tags can be extended by means of the *@itemscope* and *@itemprop* attributes, to provide extraction of RDF triples from HTML5 documents. Although the current proposal appears limited in terms of annotating with XML literals and assigning data types to annotation values, the current microdata working draft includes algorithms for direct transformations from HTML5 to RDF, as well as hints at methods to map them to the most widespread custom annotation formats, or microformats [ALLS07] (c.f. http://www.microformats.org). Along with markups for embedding knowledge, similar methods have been devised for embedding transformation rules for obtaining RDF graphs out of XML documents. For example, **GRDDL** (Gleaning Resource Descriptions from Dialects of Languages) [CON07] is a W3C Recommendation of such a technique, which consists of referencing transformations in standard elements such as the HTML link. A thread of ongoing research focuses on conventions that use GRDDL transforms for straightforward conversion across RDFa and microformats (c.f. [ADI08]).

DAML+OIL and **OWL** (Web Ontology Language) extend RDFS and emphasize support for richer logical inference (c.f. http://www.w3.org/TR/owl2-overview/). These ontology languages provide a rich set of constructs based on model theoretic semantics [HAYE04] [SCHH04]. OWL is a family of languages that allow knowledge engineers to model domains according to the principles of Description Logics (DL). This led to the definition of three fragments of the first version of OWL (OWL-Lite, OWL-DL, and OWL-Full) with varying trade-offs of expressivity and decidability [MVH04]:

OWL-Lite is the simplest variant of building a basic frame system (or an object-oriented database) in terms of class, property, subclass relation, and restrictions. OWL-Lite does not use the entire OWL vocabulary, while certain OWL terms are used under restrictions.

OWL-DL is grounded on DL and focused on common formal semantics and inference decidability. DL offers additional ontology constructs (such as conjunction, disjunction, and negation) besides class and relation, and has two important inference mechanisms: subsumption and consistency. Horrocks and Sattler in [HOSA02] argued that basic inference in most variations of DLs is decidable with complexity between polynomial and exponential time.

OWL-Full is the most expressive version of OWL but it does not guarantee decidability. The biggest difference between OWL-DL and OWL-Full is that class space and instance space are disjoint in OWL-DL but not in OWL-Full. That is, a class can be interpreted simultaneously as a set of individuals and as an individual belonging to another class in OWL-Full. The entire OWL vocabulary can be used without any restrictions in OWL-Full.

A recent revision of OWL, known as OWL2, relies on a stack of representational schemas that cover multiple layers of Tim Berners-Lee’s well-known Semantic Web layer cake. OWL2 became a W3C recommendation in late 2009, leaving room to the definition of three language profiles, such as OWL RL, OWL EL and OWL QL (c.f. OWL2: http://www.w3.org/TR/2009/REC-owl2-overview-20091027/). These subsets of OWL2 pose several restrictions on OWL language constructs and axioms in order to address certain scalability requirements deriving from interoperability with rule languages and relational databases.

Finally, to retrieve data from RDF/OWL ontologies, the **query language SPARQL**[10] has been developed. SPARQL is based on the expression of triple patterns for retrieving RDF data, and produces result sets, or RDf graphs, as output. Because of its strong formal foundation, OWL lends itself to semantic processing by DL reasoners; for example, software engines with the ability to infer the logical consequences from a set of formal axioms belonging to the realm of DL. Some reasoners are limited to the specific DL flavor of an ontology. Others can derive inferred taxonomies and arbitrary predicates that hold implicitly for general TBoxes (subsumption, satisfiability, and classification) and ABoxes (retrieval, conjunctive query answering).

Semantic Web technologies brought a set of tools for annotation, search, ontology editing, semantic wiki, semantic indexing, and so on. Early ontology engineering environments include tools such as SWOOP[11], Protégé ontology editor[12] and OntoEdit[13]. Recent ontology tools for knowledge modeling include NeOn toolkit[14] and TopBraid Composer[15]. Most of these tools rely on a plugin-based architecture and supports advanced features such as visualization, reasoning, and query. For populating ontologies, semi-automatic support based on information extraction, have been proposed; for example, AktiveDoc [LCP05]. An example of a manual approach

[10] SPARQL: http://www.w3.org/TR/rdf-sparql-query/

[11] SWOOP: http://code.google.com/p/swoop/

[12] Protégé: http://protege.stanford.edu/

[13] A predecessor to OntoStudio: http://www.ontoprise.de/en/home/products/ontostudio/

[14] NEON toolkit: http://neon-toolkit.org

[15] TopBraid Composer: http://www.topquadrant.com/products/TB_Composer.htm

is the Protégé plugin for annotating PDF documents, which is based on an ontology that is embedded in the document itself [ERI07].

2.4 Semantic Web Success Stories

Sharing success stories and best practices is necessary step to provide the first insight into these technologies and make them fit better to both the real expectations of developers and real user's needs. For example, the BBC (and other media actors) are not solely using Semantic Web technology for direct, technical advantages [SW.COM10]. The calculated guess is that just as the hyperlink revolutionized digital content distribution, the semantic hyperlink and URI promises will an even greater impact. Therefore, this section shows several Semantic Web-related success stories such as SWAN (Semantic Web ANnotator), the BBC World Cup Football 2010 Portal, the BBC London 2012 Olympics platform, the BestBuy's use of GoodRelations/ RDFa Markups. Based on a work of W3C Semantic Web Best Practice and Deployment (SWBPD) Working Group, as well as work summarized in [KW-D1.4.2v2], we present several Semantic Web best practices, case studies and use cases.

2.4.1 SWAN: Semantic Web ANnotator

SWAN is designed to perform large-scale ontology-based Information Extraction (IE) for the Semantic Web, annotating vast amounts of documents from the Web with semantic information (inferred metadata) [KW-D1.4.2v2]. SWAN is based largely on KIM [POP02a], which provides indexing, disambiguation and storage components, as well as some of the interface components. It contains two crawler versions: an HTML crawler which directly accesses web pages according to a defined scope, and an RSS crawler which uses the syndication mechanism of RSS 1.0 newsfeeds. The web pages found are then passed to the IE component, which consists of a set of processing resources implemented using GATE [CUN02b]. This pipeline of resources performs preprocessing tasks such as tokenization and sentence splitting, followed by high-level pattern matching and co-reference resolution, which results in a set of semantic annotations linking the text with concepts from an ontology. The disambiguation component performs two tasks: first, it co-refers different mentions of the same instance at the document level, and second, it continuously checks if new instances found are identical to previously found entities in other documents. Finally, the results are stored in various databases. Entities, relations and their properties are stored in an RDF Knowledge Repository, using Sesame. An index relating the entities to their source documents is stored in a Document Store that is implemented on top of Lucene Core (c.f. http://lucene.apache.org/). The annotations themselves are stored in an Annotation Store, which is implemented as a relational database. SWAN is designed to work on specific domains with the aim to improve the accuracy. However, it is also deliberately designed to be scalable, and new domains are being continuously added.

2.4.2 BBC World Cup Football 2010 Portal and London 2012 Olympics Platform

The BBC World Cup 2010 Portal (c.f. http://news.bbc.co.uk/sport2/hi/football/world_cup_2010/default.stm) collects over 700 aggregation pages (index pages), designed to lead users on to the thousands of story pages and content [RAY12]. Examples of index pages range from the Groups and Fixtures page through to detailed pages for each team or player. Previous search technologies and methods for automation and retrieval of these pages have proven to be inaccurate, e.g. it was difficult to avoid getting content mixed up between different players with the same surname. Therefore, BBC chose Semantic Web technologies for analyzing content and deciding how to tag this content with precise metadata linked to uniquely identified concepts.

The BBC World Cup 2010 Portal is arguably the first large scale, mass media site that uses concept extraction, RDF and a triple store to deliver content. It demonstrated that this kind of technology is ready to deliver large scale, mainstream products. BBC developers identified several practical advantages of using such technologies [RAY12]:

- the developers stress *flexibility* as the first reason why they used Semantic Web over more traditional technology. The flexibility played both on the data layer, where it "facilitates agile modeling" and allowed for increased query complexity compared to relational schema databases, as on the presentation layer. With regards to the presentation layer, the developers stressed that they *"are not publishing pages, but publishing content as assets which are then organized by the metadata dynamically into pages, but could be re-organized into any format we want much more easily"*;
- a second technical advantage mentioned is *inference*. Due to the reasoning facilities of the triple store, inferred statements are automatically derived from the explicitly applied journalist metadata concepts. This made both the journalist tagging and the triple store powered SPARQL queries simpler and indeed quicker than a traditional SQL approach;
- finally, *dynamic aggregations* based on inferred statements in turn increase the quality and breadth of content across the site.

The BBC Olympics 2012 Platform is based on a Dynamic Semantic Publishing (DSP) architecture, which uses LOD technology to automate the aggregation, publishing and re-purposing of interrelated content objects, according to an ontological architecture, providing a greatly improved user experience and high levels of user engagement [RAY12]. The DSP architecture curates and publishes HTML and RDF aggregations based on embedded Linked Data identifiers, ontologies and associated inference. RDF semantics improve navigation, content reuse, repurposing, search engine rankings, journalist determined levels of automation ("edited by exception"). In other words, the DSP approach facilitates multidimensional entry points and a richer navigation. The number of automated pages managed by the DSP architecture is well in excess of ten thousand, which is impossible to manage using a static CMS driven publishing stack.

A horizontal navigation through the BBC Sport website's is powered by specifically designed content model. That model links ontology concepts to navigation entries, which allows navigating to and automatically aggregating content from navigation. In other words, it allows a journalist to correctly disambiguate concepts such as football players or geographical locations. Journalist-published metadata is captured and made persistent for querying using the RDF metadata representation and triple store technology (i.e. OWLIM-SE[16], which is used for handling massive volumes of data and for intensive querying activities). The underlying navigation data and associated content model are stored within a new addition to the DSP architecture, which is a highly scaled and high performance fault tolerant Big Data Store namely MarkLogic (c.f. http://www.marklogic.com/).

Sports statistics provided by third party suppliers are stored as XML content within the Content Store. The BBC sports site queries these XML fragments adds value and re-formats the statistics in a form consumable on the sports website. The Content Store has been scaled to handle ingesting many thousands of content objects per second, whilst concurrently supporting many millions of dynamic page renditions and impressions a day.

Figure 1 shows the DSP architecture that combines SPARQL/XQuery, RDF store, and XML Store.

[16] OWLIM Edition webpage: http://www.ontotext.com/owlim/editions

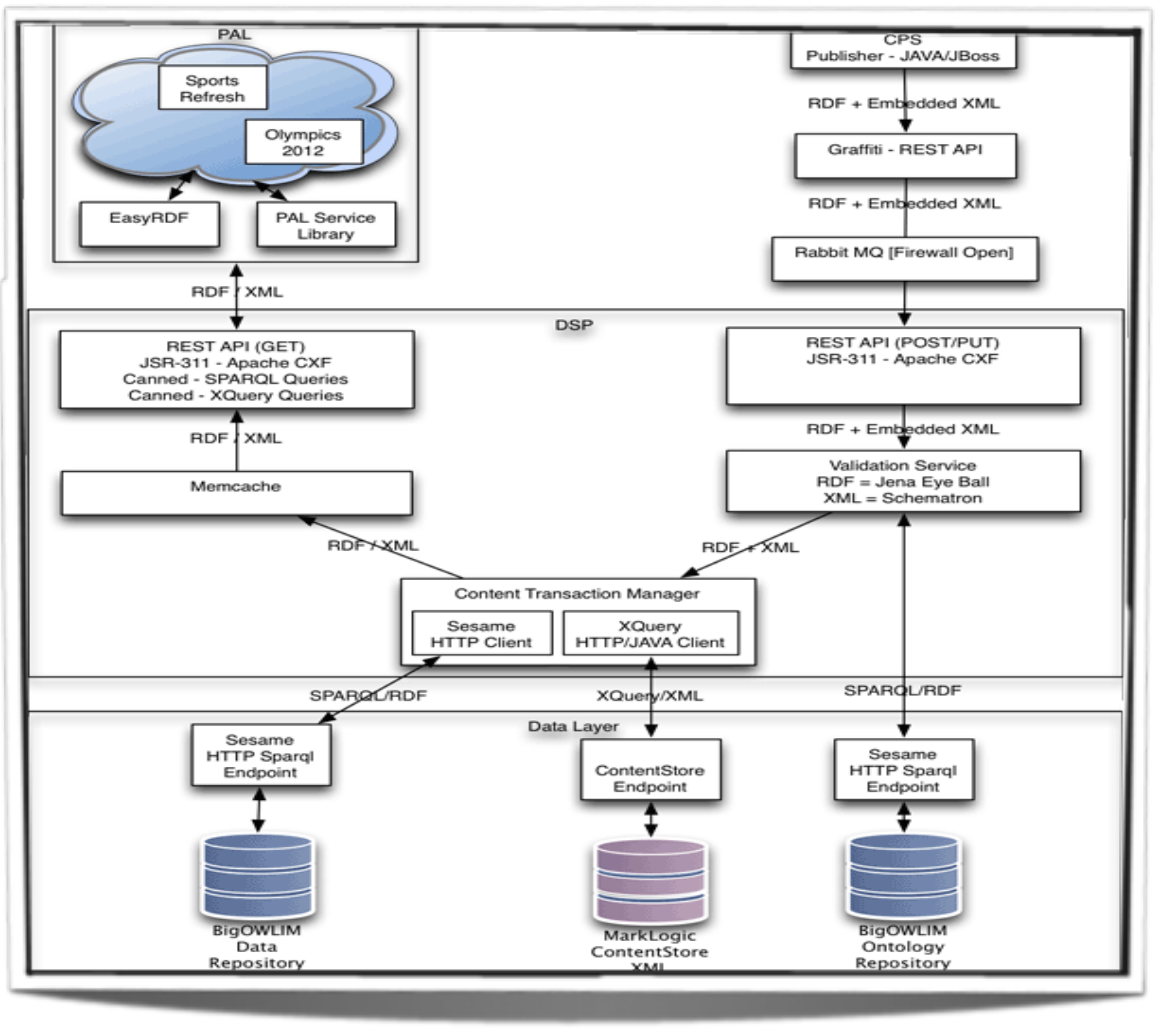


Figure 1: DSP architecture combining SPARQL/XQuery, RDF store, and XML Store.

This way, a technical architecture that combines a document/content store with a triple-store proves an excellent data and metadata persistence layer for the BBC Sport website. Replacing a static publishing mechanism with a dynamic request-by-request solution that uses a scalable metadata/data layer removes the barriers to creativity for BBC journalists, designers and product managers, allowing them to make best use of the BBC's content.

2.4.3 BestBuy’s Use of GoodRelations/ RDFa Markup

US retailer BestBuy is using GoodRelations vocabulary [GR08] to annotate web pages with RDFa content that relates to products, stores and services [SW.COM10]. RDFa offers a standardized syntax for embedding structured data into existing static/ dynamic pages such that it can be conveniently parsed and consumed by remote software agents. Developers have claimed that embedding RDFa has lead to significant SEO (Search Engine Optimisation) benefits and increased traffic. For example [ANSW10]:

- Jay Myers, a lead developer from BestBuy.com, has claimed: a 30% increase in traffic to store pages since RDFa markup was added” [RWW10].
- Anecdotal evidence exists for Google SEO benefits, where a search for ferris bueller best buy returns results where the RDFa annotated page appears above the more established page.
- Traditional search engines, including Google, are now using RDFa to generate "rich snippets" which augment keyword results with additional information, such as ratings or location. Nick Cox from Yahoo also recently reported that augmented search results (e.g. those with GoodRelations/ RDFa) get a 15% higher Click-through-Rate (CTR) in Yahoo.

GoodRelations offers an agreed-upon vocabulary for publishing product, price, and company data in RDF. Traditional Search Engine Optimization tries to put client on top of all search results, but clearly, it can work only for one company. GoodRelations puts clients on top of Web visibility for people who are looking for exactly their products or services. BestBuy have claimed significant and tangible SEO benefits through their use of RDFa. BestBuy is not the only adopter of RDFa. Tesco has also incorporated RDFa into their online product catalog[17].

[17] Tesco online product catalog with RFDa: http://www.clothingattesco.com/men/sportswear/icat/mens-sportswear#esp_sort=pdxtmagicnumber&esp_order=desc

2.4.4 Semantic Web Case Studies and Use Cases

A collection of Semantic Web case studies and use cases is given by W3C (c.f. http://www.w3.org/2001/sw/sweo/public/UseCases/). These case studies include descriptions of systems that have been deployed within an organization, and/or are now used within a production environment. The examples range from the domain of broadcasting, healthcare, life science, public institutions, search, etc. but also include those examples of prototypes which are not currently being used in business. For example, a use case on provenance tracking and data integration, which is known as “Using Semantic Web and Proof Technologies to Reduce Errors in Radiological Procedure Orders” helps to prevent medical errors that are caused by physicians overlooking vital facts. This use case supports integration of cross-domain knowledge and data seamlessly, based on explicit and unambiguous terms expressed in ontologies. The explanations generated by proof engines provide evidence to clinicians for a decision. The approach can even provide alternative solution. The final decision is still in the hands of a clinician, but making such key information and evidence readily available is extremely important when there are such large volumes of data to consider. Consult W3C use case webpage[18] for the description of more use cases and case studies.

[18] W3C use case webpage: http://www.w3.org/2001/sw/sweo/ public/UseCases/

CHAPTER 3: IKS Methodology for Building Semantic Components into CMS

Nowadays, there exist several hundred CMSs and Knowledge Management System (KMS) providers in Europe [IKS-5.0]. Although, it is still hard to make Semantic Web to operate with CMSs, many of today's web applications are making use of structuring mechanisms such as RDFa, as a first step towards getting semantics into CMS. One of the main challenges of the EU FP7 IP IKS project, which targets small to medium CMSs providers in Europe, is to improve the status quo and bring semantic technologies to CMS vendors.

The IKS final release delivers a stack of software components extending existing CMSs by adding semantic functionality. Most of IKS software components can be used as standalone server-side application extensions, which can be integrated with existing CMSs via RESTful web service interface. The rest of this section provides description of the behavior of high level requirements covering various use cases of IKS software components. The next step describes the IKS Reference Architecture (RA) for semantic CMS, which gives an overview of new concepts providing semantic functionality of CMSs. The IKS RA integrates two technology pillars:

- the **content pillar** that is already present in existing CMS architectures, and
- the **knowledge pillar** that contains novel semantic features of CMS.

The IKS Reference Implementation (RI) results in the integration of IKS final realise, which is today known as the Apache Stanbol.

This Chapter describes each of the above mention steps of the IKS methodology for building semantic CMSs. It starts with the discussion on high-level requirements, the IKS Reference Architecture, Reference Implementation, and concludes describing the IKS service integration patterns.

3.1 High Level Requirements

3.1.1 Application Requirements

The IKS applications address a set of semantic enhancements for various CMSs. A collection of application requirements in IKS represents a result of a detailed analysis of CMSs of IKS consortium partners, such as Nuxeo, Open CMS, CQ5 and TXT) and the "Brainstorming Session on Requirements for Semantic CMS"[19] that opened the project door to CMS vendors from outside of the IKS consortium. As a result of brainstorming session, the following ten horizontal high-level requirements were identified [IKS-D2.2]:

1. Common vocabulary - This requirement is about standardizing the terminology and ensuring a common language for all semantic features of IKS, which guarantee that different CMSs have the same understanding of particular features. Examples of common vocabularies are external ontologies, taxonomies, thesauri, which have the ability to provide horizontal domain knowledge.

2. Architecture and integration - The IKS architecture follows a RESTful service approach. The IKS architecture must provide customization and exchangeability of the IKS implementation. Services must be orchestrated/ recomposed to new higher order services by reusing the existing services. Services must access information inside the data repository of the CMS.

3. Semantic lifting and tagging – The IKS applications need to support different tagging and content lifting techniques, automatically or semi-automatically extracting semantics from structured and unstructured data, making suggestions about annotations, etc. Examples of content techniques are semantic navigation mechanism through the content items; automatic generation of micro-formats; semantically enhanced rich text editor; changing the presentation model based on semantic data; automatic categorization, similarity search, similarity detection,

[19] "Brainstorming Session on Requirements for Semantic CMS" website: http://www.iks-project.eu/news-and-events/press-releases/iks-requirements-workhsop-talking-community

visualization of the annotations, semantic history of navigation, providing APIs for extracting ontology from unstructured data, and more.

4. Semantic search and semantic query - The key outcomes of semantic enhancements of CMSs can be observed through semantic query and search functionality of the system. Semantic description of content has ability to improve search capabilities and provide better search results. Several sub-requirements came along with that; for example: distributed querying, support for disambiguation of search, user-friendly RDF querying, a prototype search engine understanding microformats, etc.

5. Reasoning on content items - The important requirements of IKS horizontal services is also extraction of implicit set of data from the explicit information, residing in the content repositories. In addition, IKS horizontal services need to support semantic consistency checking in CMSs.

6. Links/relations among content items - Besides semantic tagging, content items might be linked among each other. This process can be automated by using algorithms that reason on the provided tags and ontologies. As linking of content items is already a standard technique in CMS, the IKS should provide novel mechanisms to support automatic link creation, instance linking, linked data cloud, etc.

7. Workflows - Existing CMSs already provide mechanisms to represent and manage the workflows of content. The expectation of semantic CMSs is to support the handling of content workflows by using semantic information associated with the content. For example, semantic information can be used to determine the current state of a specific content in a given workflow. The IKS should provide workflows for semantic actions, which are similar to content workflows. In addition, it should support customizable workflows, intelligent content workflows that are configured based on workflow organization and hierarchy, etc.

8. Change management, versions and audit - Existing CMSs already provide mechanisms to support tasks such as change management, versions and audit.

Hence, the IKS features should be aware of content changes and provide solutions to validate semantic data. It should provide change management notifications, mechanisms for change tracking, trust management, role management, revision of content, policies for accessing the data user authentication, etc.

9. Multilingualism - The IKS semantic services should be aware of content in different languages and provide functions to reason about information even if they are created in different languages.

10. Security - Access to the content should be configured by using fine grade access control, e.g. flags such as "reasoning-allowed" or "linkable-with", instead of traditional "read-only" or "no-deletion". IKS should also support integration of permissions, roles and group models, policies for accessing the data, user authentication; roles management, trust management, etc.

3.1.2 Ambient Intelligence Requirements

High level requirements for the Ambient Intelligence (AmI) use case includes the following [IKS-D4.1]:

1. Device Input/Output Management - This requirement is about managing all technical operations on devices that are integrated inside the AmI environment. For example, it enables playback of different forms of contents (e.g. audio, video, images, speech) on the selected devices, as well as an interpretation of sensor inputs (e.g. distance or person recognition sensors). It allows other modules to get informed about interaction events and offers an interface to commit content objects for presentation on the device.

2. Device Integration - This requirement enables discovery and integration of input/ output devices, which are present within the AmI environment. It also provides the reverse process – de-installation of devices that are removed from the environment.

3. Context Management - This requirement enables management of the context that is present inside the AmI environment. This includes management of a physical situation in the AmI environment, as well as management of available devices and their users. It bridges the physical devices, which are handled by the Device Input/Output Management, and the situational knowledge, which is managed by the Situation Management. It continuously checks for context changes, and informs the Situation Management and the Knowledge Repository if changes occur.

4. Situation Management - It manages the situational part of the knowledge representation, which describes all possible situations that can occur in the AmI system. It covers the following aspects: (i) identification of relevant propositional Conceptual Models (CMs) (by using determination methods such as fuzzy search), (ii) processing of propositional CMs, (iii) alignment of propositional CMs and current situation (i.e. modifying, comparing, creating), (iv) generation of new propositional CMs, and (v) management of situation adjustment based on interaction messages.

5. Speech Communication - It enables speech-based communication between the AmI system and the user. It covers tasks such as: (i) speech input interpretation, (ii) discourse management, and (iii) dialog management. It receives speech input from the Device Input/Output Management, retrieves content that is required from the Knowledge Repository and sends presentation recommendations as the result back to the Context Management.

6. **Content Retrieval and Knowledge Extraction Pipeline** – This requirement retrieves content objects from external (unstructured) sources and integrates them within the AmI environment. It performs the following tasks: (i) content aggregation, (ii) content reengineering that transform retrieved content into an ontological representation, (iii) content refactoring that maps the external content concepts into AmI concepts and (iv) content filtering. The content refactorer stores content items into the Knowledge Repository as an external knowledge. The binary parts of some content items (e.g. audio content) are stored in the Content Repository.

3.2 IKS Alpha: Refactoring CMS and Semantic Web Technology

The development process in IKS combines two approaches: top-down design and bottom-up prototyping. The IKS Alpha is the first software release in the project that brought together the results of both approaches top-down and bottom-up.

The bottom-up approach was driven by the industrial partners and their requirements with respect to semantic add-ons to their existing CMSs. The first project meeting discussing a set of industrial requirements around semantically enhanced CMSs, created a very basic infrastructure for further development of the IKS semantic enhancer. Such enhancer is called FISE (Furtwangen IKS Semantic Engine). Beside FISE, several other components evolved, such as INTERPRET; a set of FISE services that address semantic lifting requirements; KReS services that address models such as ontology, management functionalities; a Persistence Store which uses different triple store implementations as its backend (Figure 2).

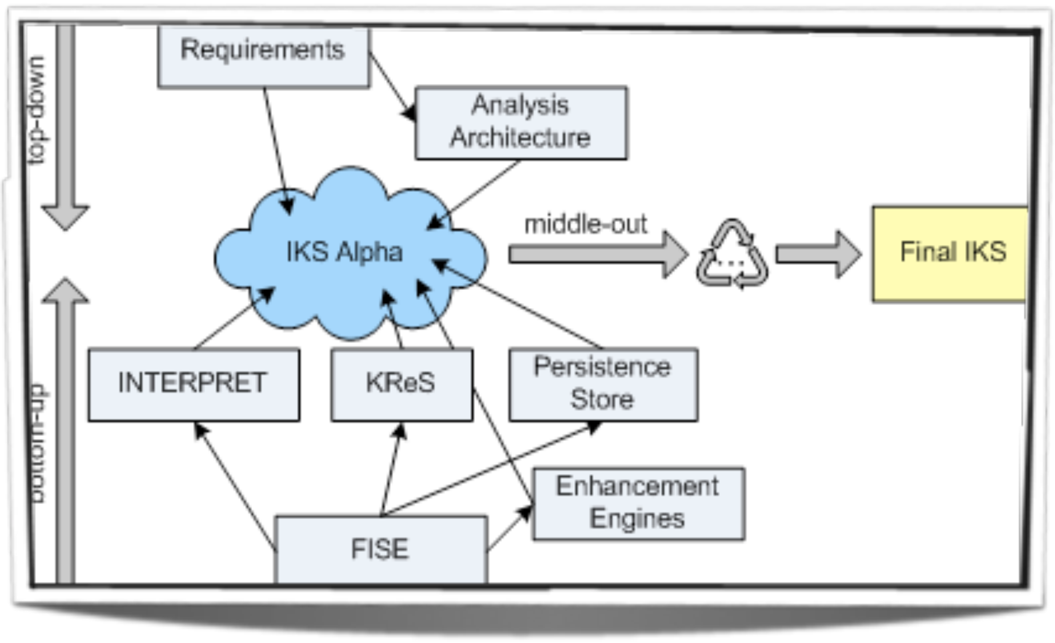


Figure 2: The IKS Development Process Overview

3.2.1 Semantic Engine (FISE)

The idea of FISE [FISE] is to create a semantic engine that is accessible through a simple RESTful HTTP interface. Therefore, the FISE engine was build to support the enhancement of content with semantic meta-data, as shown in Figure 3.

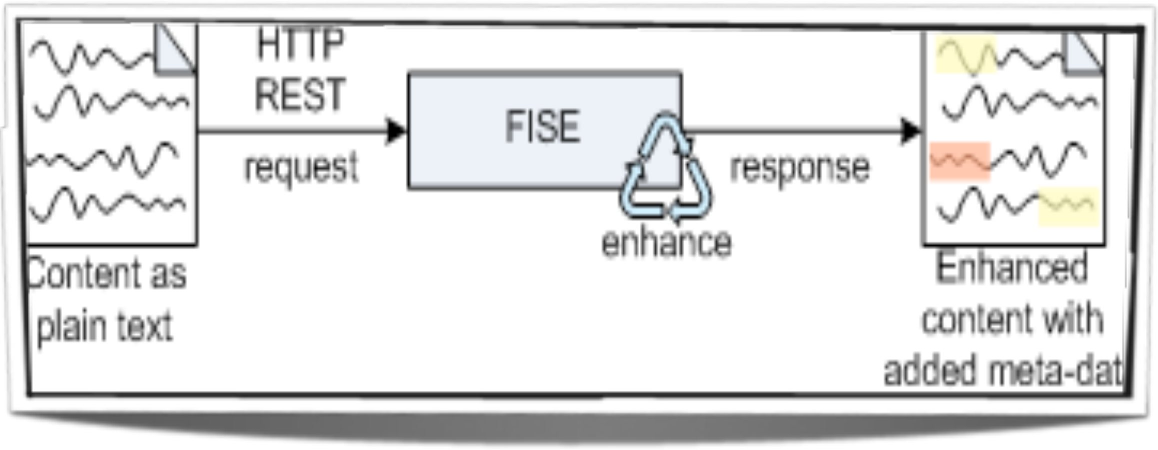


Figure 3: The FISE Approach

The FISE architecture consists of three parts: (i) a Job Manager that receives incoming requests for enhancement of the content and delegates them to (2) enhancement engines. In addition, FISE can (3) store the content along with the extracted meta-data.

Each component of FISE is an OSGi bundle. At runtime, the different bundles are linked and act together. OSGI also supports distributed component development: once the interfaces between components are defined, each component can be designed and implemented independently of others.

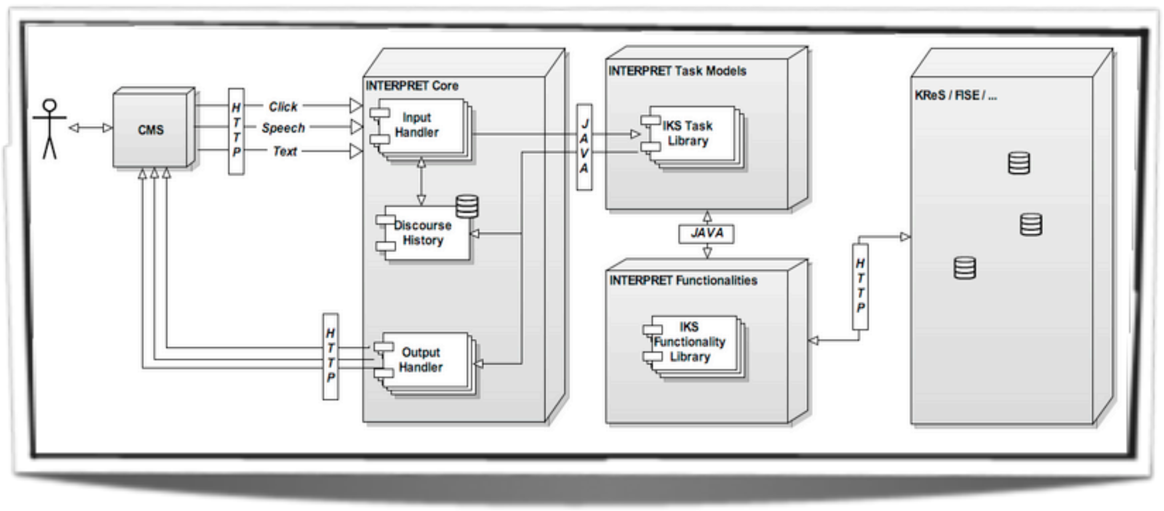


Figure 4: The INTERPRET component

3.2.2 Knowledge Interaction (INTERPRET)

The INTERPRET component is responsible for managing and supporting interaction of users with the content. In INTERPRET, the user's interaction is seen as task that can be further modeled (e.g., contribute content to the CMS, retrieve content...). The task models in INTERPRET can trigger various functionalities, such as retrieve related content, etc. INTERPRET can be customized at any time, by adding new task models or new functionalities to the system. It can also create recommendations on how, when and which content should be presented (Figure 4). More technical details on INTERPRET can be found in [IKS-D5.1].

3.2.3 Knowledge Representation and Reasoning (KReS)

The Knowledge Representation and Reasoning System (KReS) is a standalone set of software components targeting several functionalities and requirements belonging to the IKS knowledge management layer [IKS-D3.2].

KReS Alpha is developed in Java as a set of OSGi components for the Apache Felix platform. KReS provides developers with a Java API and a set of RESTFul services. It relies on the OWLApi for the ontology management (e.g., OWL2 and OWLLink support), on the Jena API for RDF-related features (e.g., SPARQL support), and also contains Hermit[20] as a built-in reasoner.

KReS contains five main software components, each consisting of an OSGi bundle. KReS components are shown in Figure 5, and described below.

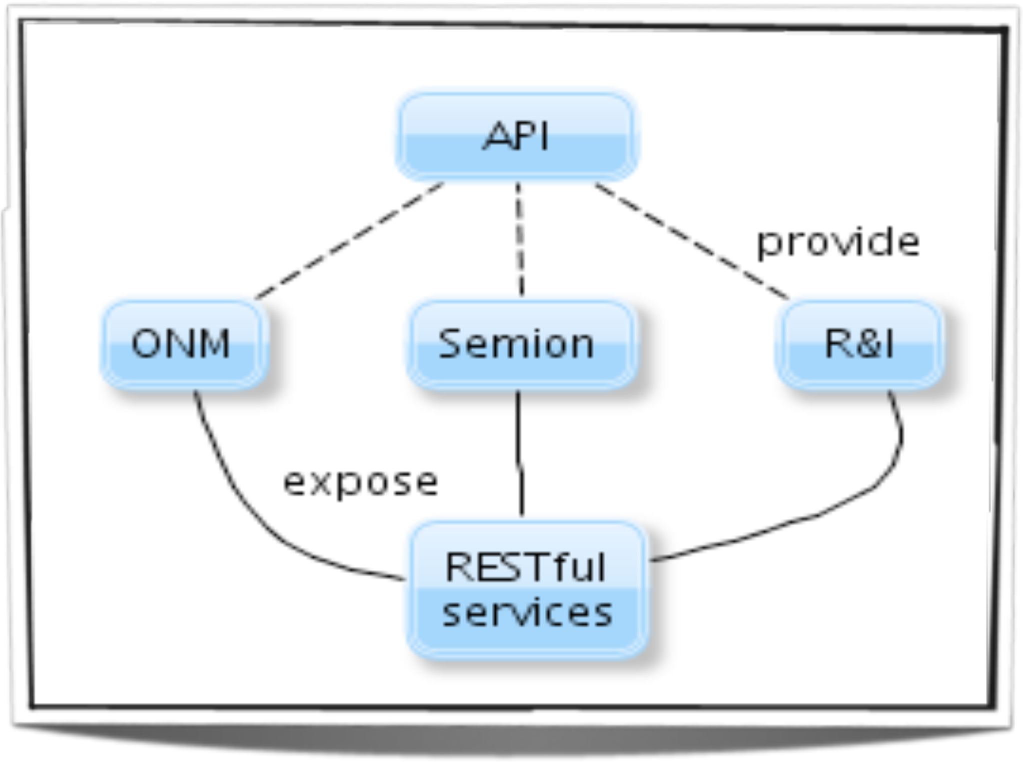


Figure 5: KReS components

- API: It provides interfaces and abstract Java specifications of all components that are intended to be of interest for CMS developers. The API is documented

[20] Hermit reasoner website: http://hermit-reasoner.com/

as a reference for all developers who wish to provide custom implementations, or interact with KReS programmatically;

- RESTFul services: It provides a set of HTTP RESTful services for using KReS functionalities from client applications;
- Ontology Network Manager (ONM): It implements the API for managing OWL (including OWL2) ontologies, in order to prepare them for usage by reasoning services, refactorers, rule engines and the like;
- Rule Manager and Inference Engine (R&I): It implements the API for the management of rules, and the execution of rule sets (called recipes in KReS), and the execution of reasoning tasks;
- Reengineering and Refactoring Engines (SEMION): It provides a set of functionalities for reengineering and refactoring of models, e.g., triplification, performed over the knowledge stored in a CMS persistence store, according to a set of customized ontologies and rules.

More details about KReS are available in [IKS-D5.2] .

3.2.4 Persistence Store

Persistence Store provides storage and access points for the semantic data. It uses two types of interfaces such as Java interfaces and REST interfaces. As an OSGi bundle, Persistence Store implements those interfaces that are specified by FISE. Figure 6 shows architecture of the Persistence Store component. The description of the detailed architecture and implementation details are available in [IKS-D5.4].

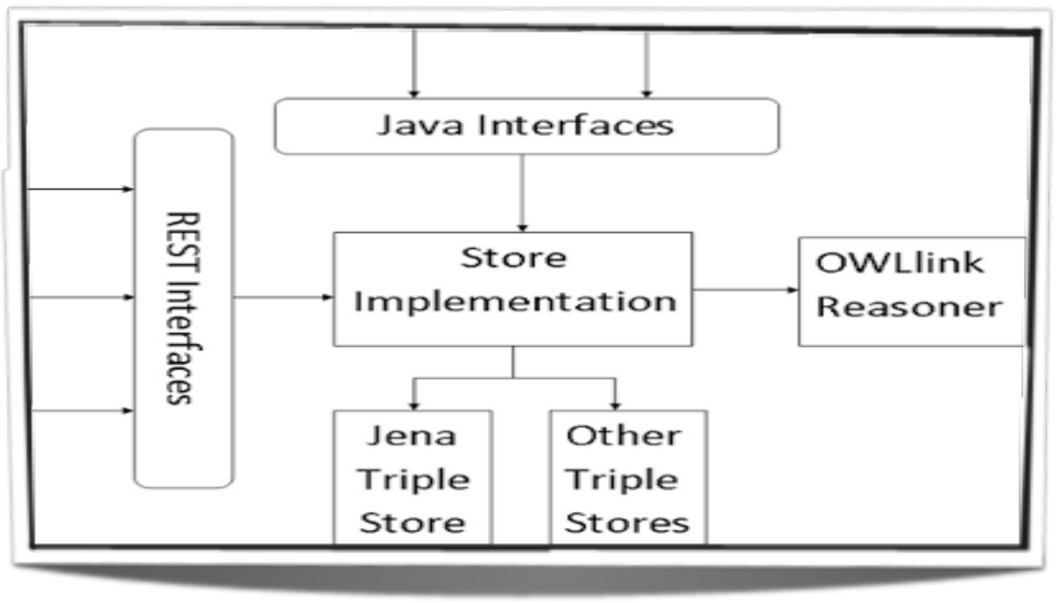


Figure 6: Persistence Store Architecture

3.3 The IKS Reference Architecture

Figure 7 shows two major parts of the IKS reference architecture [IKS-D4.2].: The left side of the reference architecture includes all features, which are required to handle content, while the right side adds semantic features. The interface features connect these two sides via a traditional user interface.

Furthermore, the right side of the IKS reference architecture is divided into the four main layers, such as (i) Presentation and Interaction layer, (ii) Semantic Lifting layer, (iii) Knowledge Representation and Reasoning layer, and (iv) Persistence layer. These four layers are further refined into a set of feature layers. Each feature layer encapsulates required features at the different high-level layers for a semantic CMS. Possible combination of features from each layer will be described by means of so-called IKS service integration patterns (see Section 3.5).

Table 1 briefly describes each of the feature layers.

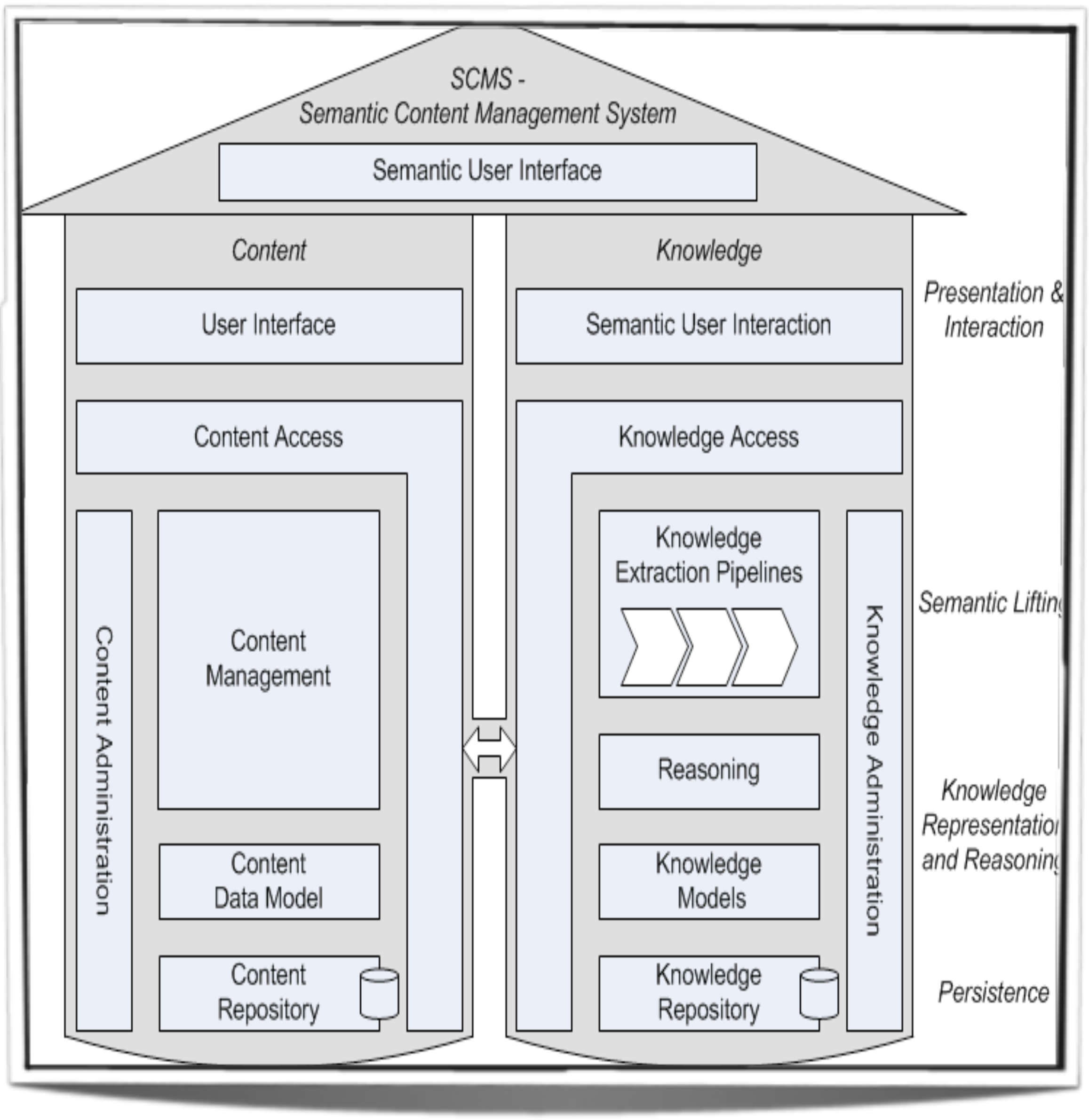


Figure 7: IKS Reference Architecture for a Semantic CMS

Feature	Short Description
Semantic User Interface	A semantic user interface uses available semantic metadata and, based on the provided information, adapts its behavior.
Semantic User Interaction	A semantic context of user interaction is provided by the semantic user interaction features that control overall user interaction with the system.
Knowledge Access	A knowledge access needs to ensure a standardized access to all participating services within the knowledge column.
Content Integration	It bridges two sides of the IKS reference architecture by integrating existing content from the CMS with its semantic enhancements.
Knowledge Extraction Pipelines	Different knowledge extraction pipelines are used to extract different semantic metadata from the content.
Reasoning	Based on the available semantic metadata and the defined knowledge models, it is possible to infer new knowledge by following semantic relations. Automatic reasoning features are used to evaluate the available metadata in combination with the knowledge models.
Knowledge Models	Knowledge models are used to internally represent the semantic metadata and define the available semantic relations. Such knowledge models are often defined in terms of an ontology.
Knowledge Repository	In contrast to a content repository, the knowledge repository is optimized for storing semantic metadata and its relations.
Knowledge Administration	The different knowledge features need to be administered and configured to be used in different usage scenarios. Each feature has to provide an administration interface, which is bundled in a centralized administration console to configure the whole stack.

Table 1: Features of the IKS Reference Architecture and their descriptions

In the following, we introduce the IKS reference implementation for the above reference architecture.

3.4 The IKS Reference Implementation

The IKS reference implementation is an instance of the IKS reference architecture. The IKS reference implementation is based on two open-source projects, such as VIE and Apache Stanbol (Figure 8).

The objective of VIE is to ease the development of semantic web applications on user interaction level. VIE's service-architecture provides communication to different backend services directly from the browser. For example, the main features of the VIE Satnbol service are text enhancement, lookup for entities in the Entity Hub, getting metadata for specific entities, and storing entities that were created or changed during user interaction. Hiding the complexity of the back-end engines is essential for building long-lasting front-end applications, which are also more resistant against future changes of the different back-end components. VIE is designed to be a JavaScript framework that can be extended to implement custom user interface widgets. It can use the Apache Stanbol RESTful API for implementing semantic interaction in web applications. VIE can work together with any CMS with the goal to decouple CMS and user interface.

The second project implementing the IKS reference architecture is Apache Stanbol, which is focused on the development of flexible services on the server-side of a semantic CMS. The Apache Stanbol' components implement the knowledge side of the IKS reference architecture starting from the Knowledge Access layer (as shown in Figure 8).

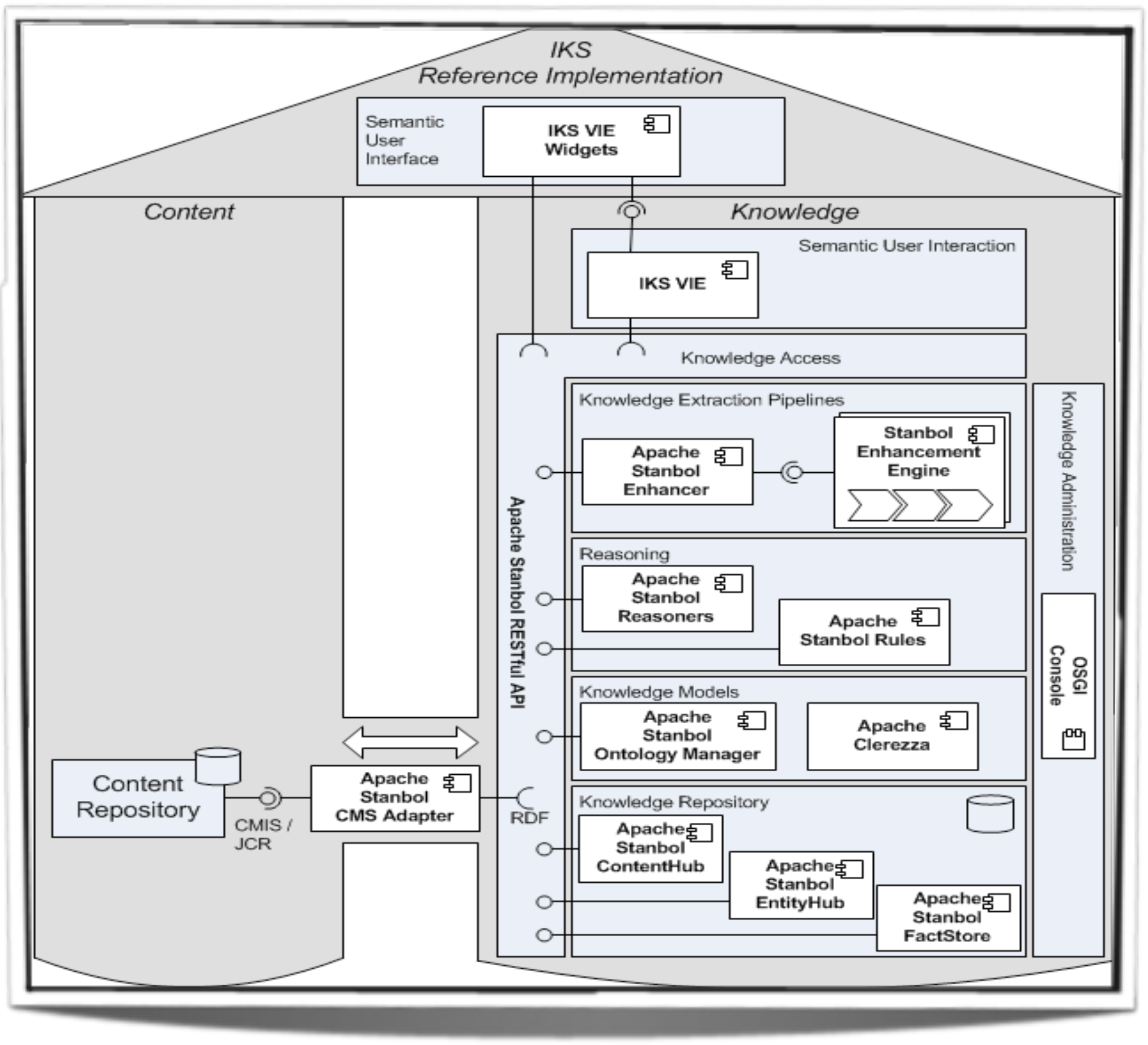


Figure 8: The IKS Reference Implementation

The Apache Stanbol project was “incubated” in November 2010 with the aim of supporting an open-source community adopting semantic technologies for CMSs. The FISE component of the IKS Alpha was migrated to Apache Stanbol. Since then, most of the IKS software was directly developed under Apache Stanbol and is freely available. In other words, Apache Stanbol is no longer driven by the IKS project consortium members, but attracts attention of independent open-source developers who also contribute to its development. In September 2012, Apache Stanbol graduated to a full project of the Apache Software Foundation.

Apache Stanbol is a modular set of components and HTTP services for semantic CMS. It extends traditional CMS with features for semantic content enhancement. It provides the following components, which are listed in Table 2.

Component	Short Description
Apache Stanbol Enhancer	The Enhancer and its Enhancement Engines (formerly known as the FISE component) are the components aimed to enhance given content with additional semantic metadata.
Apache Stanbol Reasoner	Reasoner is used for gaining additional knowledge by following the semantic relations defined in the knowledge base. An example is to retrieve the additional knowledge that Bob is grandfather of Kate by knowing that Pete is son of Bob and father of Kate.
Apache Stanbol Rules	Inference Rules, also known as transformation rules, are syntactic rules which take premises and return a conclusion. These rules can be used to transform the metadata into other vocabularies, etc.
Apache Stanbol Ontology Manager	Ontologies are used for defining the knowledge models that describe the metadata of content. Additionally, the semantics of your metadata can be defined through an ontology. The reasoners and rule features are based on such ontology definitions.

Component	Short Description
Apache Stanbol Content Hub	The Content Hub is the component which provides persistent document storage whose back-end is the Apache Solr. It enables semantic indexing during text-based document submission.
Apache Stanbol Entity Hub	The Entity Hub is the Apache Stanbol component which deals with entities and their metadata. It is a generic component that is able to connect to a configurable list of open-linked databases, enriching information about entities from various sources.
Apache Stanbol Fact Store	The Fact Store is used to store relations between entities. It only uses references to entities via their URI. The entities should be handled by the Entity Hub.
Apache Stanbol CMS Adapter	The CMS Adapter component acts as a bridge between JCR/CMIS compliant CMSs and Apache Stanbol. It is used to map existing node structures from JCR/CMIS content repositories to RDF models, or vica versa.

Table 3: Components of the Apache Stanbol and their short description

CHAPTER 4: Apache Stanbol and VIE in a Semantic CMS Technology Stack

The major components of the IKS Semantic CMS Technology Stack are implemented through IKS foundational components, such as Apache Stanbol, its software components and services[21].

4.1 Foundational Components of IKS

Apache Stanbol is designed to extend existing CMSs with semantic services. It can be also used to tag extraction/suggestion, text completion in search fields, smart content workflows, email routing based on extracted entities, topics, etc.

Apache Stanbol is built as a modular set of components (shown in Figure 9). Each component is accessible via its own RESTful web interface. All components are implemented as OSGi bundles, components and services. By default Apache Stanbol uses the Apache Felix OSGi environment. For deployment, it uses the Apache Sling launcher, and can be run as a standalone application or as a web application that is deployable in servlet containers such as Apache Tomcat.

[21] http://www.iks-project.eu/sites/default/files/iks_del_5_0_design_implementation_IKS_compendium_2012.pdf

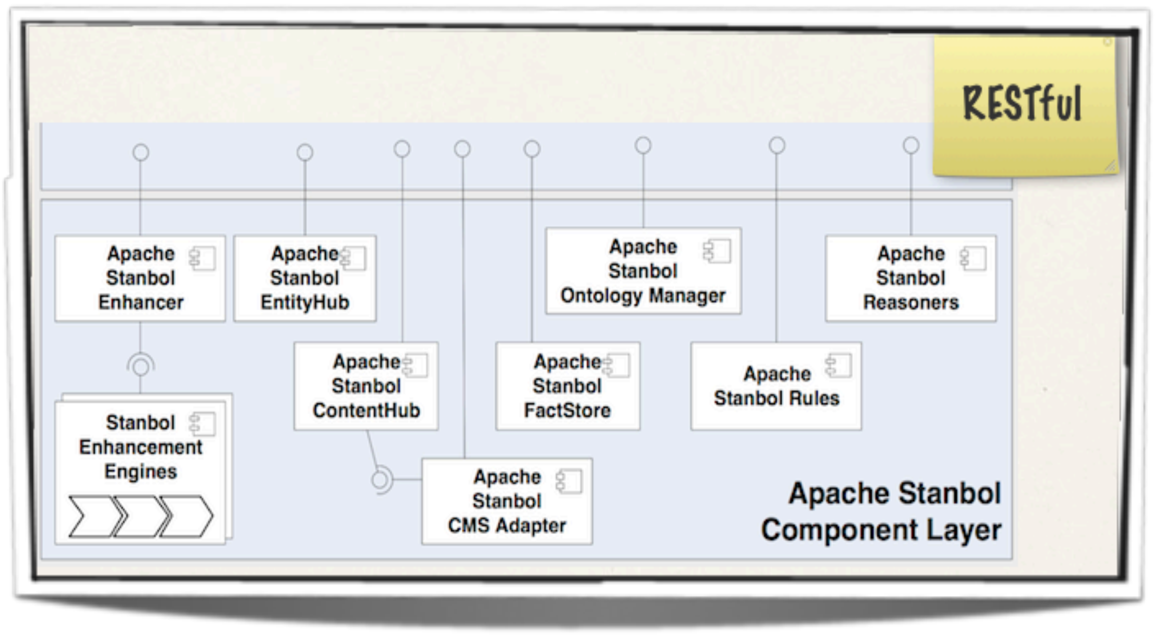

RESTful
Apache Stanbol Enhancer
Apache Stanbol EntityHub
Apache Stanbol Ontology Manager
Apache Stanbol Reasoners
Stanbol Enhancement Engines
Apache Stanbol ContentHub
Apache Stanbol FactStore
Apache Stanbol Rules
Apache Stanbol CMS Adapter
Apache Stanbol Component Layer

Figure 9: Apache Stanbol Components

The main features of Apache Stanbol are:

- **Content Enhancement:** Services that add semantic information to "non-semantic" pieces of content;
- **Reasoning:** Services that are able to retrieve additional semantic information about the content based on the semantic information retrieved via content enhancement;
- **Knowledge Models**: Services that are used to define and manipulate the data models (e.g. ontologies) that are used to store the semantic information;
- **Persistence:** Services that store (or cache) semantic information, i.e. enhanced content, entities, facts, and make it searchable.

In the following, we discuss Apache Stanbol software components[22].

[22] http://www.iks-project.eu/sites/default/files iks_del_5_0_design_implementation_IKS_compendium_2012.pdf

4.1.1 Apache Stanbol Enhancer

Apache Stanbol Enhancer[23] and its Enhancement Engines[24] are designed to support content enhancement. Apache Stanbol Enhancer provides both a RESTful and a Java API that allows a caller to extract features from passed content. In case of RESTful API, Apache Stanbol takes the content and delivers it to a configurable chain of enhancement engines. Each enhancement engine in this chain is used for a specific purpose. There are preprocessing engines, e.g. engines for converting the content into the correct format, engines that automatically extract semantic metadata about the content, etc. As an example, Apache Stanbol provides an engine to automatically determine the language of a text. Other engines are able to extract entities such as persons and places directly from the text. The response will hold the RDF enhancement serialized in the format that is specified in the Accept header.

In case of Java API, after the enhancement process, ContentItem do not only contains the metadata but also other information such as converted versions of the passed content. The figure below provides an overview of the RESTful as well as the Java API provided by the Stanbol Enhancer.

[23] Apache Stanbol Enhancer webpage: http://stanbol.apache.org/docs/trunk/components/enhancer/

[24] Apache Stanbol Enhancement Engines webpage: http://stanbol.apache.org/docs/trunk/components/enhancer/engines/index.html

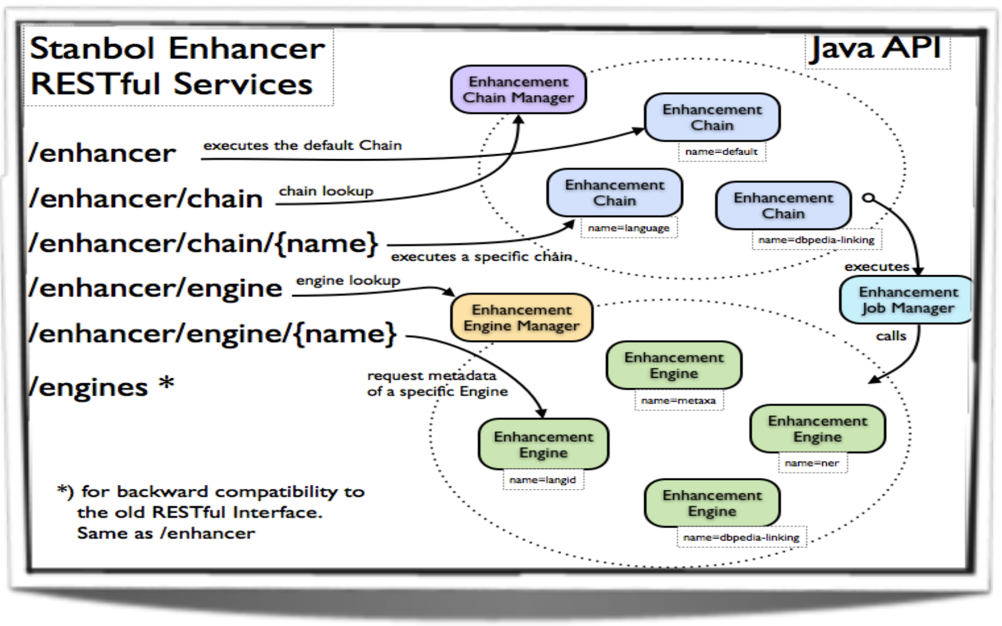


Figure 10: Stanbol Enhancer – RESTful API and Java API

The main interfaces and utility classes of Apache Stanbol Enhancer are as follows:

- **ContentItem**: A content item is the unit of content Stanbol Enhancer deal with. It gives access to the binary content that was registered, and the graph that represents its metadata (provided by client and/or generated). ContentItems are created by using the ContentItemFactory.
- **EnhancementEngine**: The enhancement engine provides the interface to internal or external semantic enhancement engines. Typically content items will be processed by several enhancement engines.
- **EnhancementChain**: An enhancement chain represents a user provided configuration, which describes how content items passed to this chain should be processed by the Stanbol Enhancer. The chain defines a list of available enhancement engines and their order of execution.
- **EnhancementJobManager**: The enhancement job manager performs the execution of the enhancement process as described in the execution

plan provided by the enhancement chain. The enhancement job manager is also responsible for recording the execution metadata.

- **ChainManager**: The chain manager allows to lookup all configured enhancement chains. It also provides a getter for the default chain.
- **EnhancementEngineManager**: The enhancement engine manager allows to lookup active enhancement engines by their name.

The enhancement structure of Apache Stanbol defines types and properties used for the resulting metadata graph of the Enhancer. The enhancement structure defines three types of annotations: text, entity, and topic annotation. Text annotation describes the occurrence of an extracted feature within the parsed text. Entity annotation suggests an entity for mention within the text (e.g. dbpedia:International_Monetary_Fund for the mention "IMF" in the analyzed Text). Topic annotations assign the parsed document (or parts of the document) to topics and categories. In addition, all annotations created by Stanbol Enhancer provide additional meta information.

The Apache Stanbol Enhancement Engines, formerly known as the FISE component, includes the following engines and their features:

- Preprocessing:
  - Tika Engine: it provides content type detection, text extraction from various document formats, extraction of metadata from document formats, etc.;
  - Metaxa Engine: it provides a generic framework for extracting plain text and embedded metadata from documents, images and audio files. A large number of standard document formats is supported in the default configuration, ranging from office documents from the major vendors, as well as standard image and audio formats. Special attention was given to HTML documents. In addition to text extraction, it supports the extraction of structured annotations embedded in HTML content, such as RDFa and microformats;
- Language detection:

- Language Identification Engine: It provides language detection for textual content by using the Apache Tika[25] software. This engine is a pre-requisite to allow other components to activate and use language specific resources;
- Language Detection Engine: It provides language detection for textual content by using a Language Detection library[26], which currently support 53 languages (c.f. https://code.google.com/p/language-detection/wiki/LanguageList);
- RESTful Language Identification Engine; CELI Language Identification Engine, etc.

- Sentence detection:
    - OpenNLP Sentence Detection Engine is based on OpenNLP;
    - Smartcn Sentence Detection Engine: it adds sentence detection support for Chinese.
- Tokenizer engines:
    - OpenNLP Tokenizer Detection Engine is based on OpenNLP;
    - Smartcn Tokenizer Engine: it adds tokenization detection support for Chinese;
    - Paoding Tokenizer Engine: it is a part of Paoding Analyzer Integration that adds tokenization detection support for Chinese;
- Part of Speech (POS) Tagging:
    - OpenNLP POS Tagging Engine: it is a POS tagger implementation based on OpenNLP;
- Named Entity Recognition (NER) Engines
    - Open NLP NER Engine;
    - OpenNLP Custom NER Model Engine;
    - OpenCalais Enhancement Engine: it provides a free high-quality online service for Named Entity Recognition and Relation Extraction in the news domain. The Apache Stanbol OpenCalais Engine provides an

[25] Apache Tika webpage: http://tika.apache.org/

[26] Language Detection library webpage: http://code.google.com/p/language-detection/

interface to that service. It also provides means for mapping OpenCalais entity categories to user specified categories;

- Named Entity Extraction Engine: This engine is based on the NLP features of Apache OpenNLP. It uses its maximum entropy models to detect persons, names and organizations;
- Linking / Suggestions
    - Keyword Linking Engine: The Keyword Linking Engine supports the extraction of keywords in multiple languages;
    - Geonames Engine: This engine creates fise:EntityAnnotations based on the http://geonames.org dataset. It suggests links to geonames.org and provides hierarchical links for locations;
    - Zemanta Engine: This is an enhancement engine with Zemanta API. A Zemanta API key is required to run this engine;
- Postprocessing / Other
    - Refactor Engine: It refactors RDF graphs of recognized entities to a target vocabulary. It transforms enhancements according to a target ontology, requires KRES launcher.

A detail list of available Apache Stanbol Enhancement Engines is available on the Apache Stanbol website[27].

4.1.2 Apache Stanbol Entityhub

Apache Stanbol Entityhub deals with entities and their metadata. It is a generic component with ability to connect to a configurable list of open linked databases. It provides information about entities relevant to the user’s domain [IKS-D4.2]. Figure 11 provides an overview of the Entityhub’ features.

[27] http://stanbol.apache.org/docs/trunk/components/enhancer/#Using_Stanbol_Enhancer

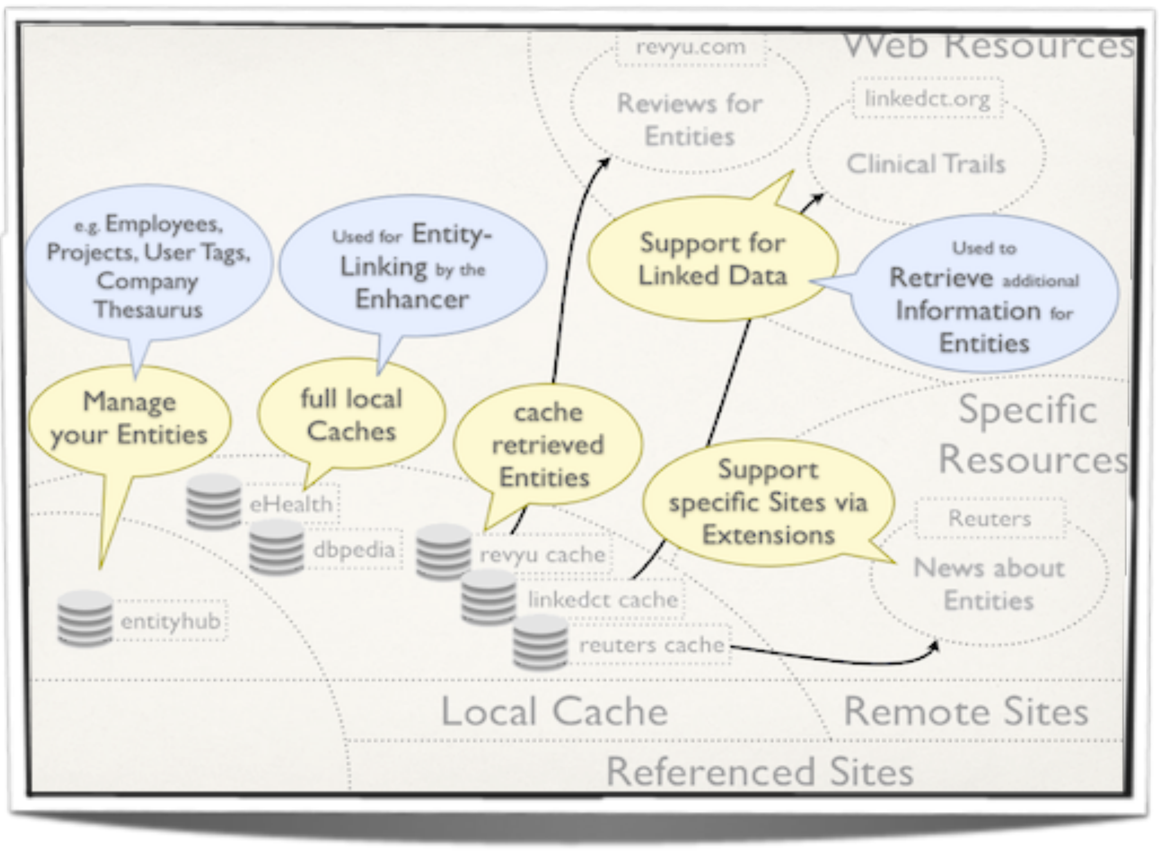


Figure 11: Features of Apache Stanbol Entityhub

The main features are the following:

- Entityhub (/entityhub): It manages local entities and import entities from websites, and defines mapping between local entities and those entities managed by websites.
- Site Manager (/entityhub/sites): It provides unified access to all currently active websites. Requests sent to these endpoints are forwarded to all active websites.
- Sites (/entityhub/site/{siteId}): Sites are entity sources that are integrated within Apache Stanbol Entityhub. There are two different types of Sites:
    - ReferencedSite: It supports local caches and indexes. A local cache allows to locally store retrieved entity data in order to speed-up retrieval,

while a local index is a locally available index over the data of the remote dataset. If such an index is available, all requests are processed using the index. Local Indexes are created by the Entityhub Indexing tool.

- ManagedSite: It allows users to manage their own entity by using the RESTful API.

4.1.3 Apache Stanbol Contenthub

Apache Stanbol Contenthub is a document repository that provides semantic storage for the content items and their semantic search services [IKS-D4.2]. Text-based documents can be submitted, semantically indexed and searched through services of Contenthub such as (i) storage services (i.e. Solr store, In Memory store, Clerezza store) and (ii) search services (i.e. related keyword search, Solr search, Featured search). All documents in the Apache Stanbol Contenthub are *content items*. Apart from the actual text-based content, a content item also includes its metadata.

Apache Stanbol Contenthub provides a default configuration of Apache Solr, which enables powerful indexing and text-based search mechanisms by indexing the documents. When the document is submitted, semantic information about the entities contained within that document are retrieved through the Apache Stanbol Entityhub, and then, enhanced along with the document. A user can also provide additional metadata related to the content item. All additional information is indexed along with the content itself. For example, if a document submitted to the Contenthub includes the keyword “Istanbul”, then the country information related to this keyword - “Turkey” and the regional information such as “Marmara”, become indexed along with this document. This leads to more accurate search results over the content items.

Search functionality of Apache Stanbol Contenthub is built on top of the Apache Solr infrastructure. Apache Stanbol Contenthub comes together with a default “semantic” index that corresponds to an Apache Solr core. Based on additional metadata, which is indexed together with the content, users can build up faceted search mechanisms. Facets are organized under categories, such as organizations, places, and people. By using faceted search facility, users can set new filters on top of the existing facets, or remove the existing filters through GUIs that allows expanding or narrowing the search scope.

Apache Stanbol Contenthub also offers related keywords extraction based on the latest search operation. Such functionality enhances navigation mechanisms by giving users another similar and/or related content items. The keyword suggestion mechanism uses three sources:

- Ontology resources: If there are ontologies which are registered within Apache Stanbol, suggested keywords will be retrieved from these ontologies;
- Referenced sites: Related keywords can be retrieved from referenced sites, which are registered within Apache Stanbol. For example, DBpedia comes as a default referenced site within Apache Stanbol.
- Wordnet: If a Wordnet database is configured within Apache Stanbol, then related keywords can be retrieved directly from the Wordnet database.

Apache Stanbol Contenthub integration with LMF and LDPath

The default semantic index of the Apache Stanbol Contenthub considers several generic semantic relations among entities. At the same time, it offers the ability of creating new Apache Solr cores, which directs the system while indexing and searching only to those content items that are important in the domain. The LMF (Linked Media Framework) project[28] provides this functionality as-is. The LMF Semantic Search module creates Apache Solr indexes via the RDF Path Language[29]. A standalone library for the evaluation of the RDF Path Language is called LDPath[30]. It is a simple path-based query language over RDF, which is particularly designed for querying Linked Data Cloud by following RDF links between resources. To support domain specific indexing, LDPath is integrated into the document submission and search processes of the Apache Stanbol Contenthub. Users are expected to create their LDPath programs beforehand, so that they can be used in submission and search operations.

[28] Linked Media Framework (LMF) webpage: http://code.google.com/p/lmf/

[29] RDF Path Language webpage: http://code.google.com/p/ldpath/wiki/PathLanguage

[30] LDPath webpage: http://code.google.com/p/lmf/wiki/ModuleLDPath

In addition, the LMF's Semantic Index Manager indexes the content. Semantic indexes are further controlled through LDPath programs, which are uniquely identified by their names within Contenthub. Each LDPath program, submitted to Contenthub, triggers a new Apache Solr core. In addition to document search over semantic indexes of Apache Solr, the Apache Stanbol Contenthub provides a suggestion (related keyword) mechanism for the query terms (given keywords). Contenthub also provides a featured search interface, which combines the capabilities of Apache Solr search and related keyword search, as well as tokenization service. The tokenization service uses Entityhub in order to extract entities within the query term and to facilitate the search according to the extracted entities.

4.1.4 Apache Stanbol Ontology Manager

When processing huge knowledge bases consisting of CMS-based data, together with external, linked open data, scalability problems can be expected to occur. The Apache Stanbol Ontology Manager provides a suite of functionalities such as retrieval, aggregation, loading and concurrent management of knowledge bases. It provides a **controlled environment** for managing ontologies, **ontology networks** and user sessions for semantic data. It provides full access to ontologies, which are stored into the Apache Stanbol persistence layer. Managing an ontology network means activating or deactivating parts of a complex model, so that data can be viewed and classified under different "logical lenses". This is especially useful in ontology reasoning.

Key functionalities of the Apache Stanbol Ontology Manager are as follows:

- Creating customized views over the entire Apache Stanbol knowledge base. This favors scalability in reasoning processes and implements the concurrent management of multiple ontology networks;
- Organizing and importing controlled vocabularies, upper ontologies, design patterns and custom ontologies into convenient ontology libraries and configuring caching policies for them;
- Interactively accessing stored ontologies and their elements.

The OntoNet component is responsible for the creation of customized views in the Apache Stanbol Ontology Manager. It allows for constructing subsets of the knowledge base managed by Apache Stanbol into OWL/OWL2 ontology networks. Organizing and importing controlled vocabularies, upper ontologies, design patterns and custom ontologies is implemented by the Ontology Registry Manager, while the Store component of the Apache Stanbol Ontology Manager provides access to stored ontologies. The OntoNet component allows for setting up and managing multiple virtual ontology networks connecting various ontologies.

4.1.5 Apache Stanbol Rules

This is a component that supports construction and execution of inference rules. An inference rule, or transformation rule, is a syntactic rule or function, which takes premises and returns a conclusion. It adds a layer for expressing business logics by means of axioms that is encoded by inference rules. Axioms can be organized into a container that is called - a recipe, which identifies a set of rules that share the same business logic and interpret them as a whole.

Rules can be expressed and processed in three different formats - SWRL, Jena rules, and SPARQL.

- Semantic Web Rule Language (SWRL) is a rule language, which combines OWL DL with the Unary/Binary Datalog RuleML sublanguages of Rule Markup Language and enables Horn-like rules to be combined with an OWL knowledge base. Providing Apache Stanbol Rules as SWRL rules means that they can be interpreted in classical DL reasoning. That allows for Apache Stanbol Rules to be used with any of the OWL 2 reasoners configured in the Apache Stanbol Reasoner component;
- Jena Rules enable compatibility with inference engines based on Jena inference and rule language. The Apache Stanbol Reasoners component provides a reasoning profile based on Jena inference.
- SPARQL is a query language for RDF. A natural way to represent transformation rules in SPARQL is by using the CONSTRUCT query form. Apache Stanbol Rules can be converted to SPARQL CONSTRUCTs and

executed by any SPARQL engine. Apache Stanbol Rules provides a particular SPARQL engine that is called Refactor, which performs transformation of RDF graphs based on transformation rules from Apache Stanbol. The latter allows, for instance, the vocabulary harmonization of RDF graphs to be retrieved from different sources in Linked Data.

Apache Stanbol Rules allows for integrity check for data fetched from heterogeneous and external sources, to prevent unwanted formats and inconsistent data. It also provides information integration and vocabulary harmonization for different semantically enhanced contents.

4.1.6 Apache Stanbol Reasoner

The Apache Stanbol Reasoner component provides a set of services that take advantage of automatic inference engines. This module implements a common API for reasoning services, providing the possibility to perform different reasoners in parallel. This module includes OWL API and Jena-based abstract services, which provide implementations for Jena RDFS, OWL, OWLMini and Hermit reasoning service. The Apache Stanbol Reasoner can be used to automatically infer additional knowledge and obtain new facts in the knowledge base.

4.1.7 Apache Stanbol FactStore

The Apache Stanbol FactStore component lets a user store relation between two or more entities called - *a fact* [IKS-D4.2]. FactStore stores only the relations between entities, while Entityhub handles entities themselves. Fact Store is used to store N-ary facts according to a user defined fact schema.

A fact schema can be defined by specifying roles with corresponding types of entities, which take a part in the fact. It provides a simple way to define and store facts, and can be used in those scenarios which do not required a complex ontology to be defined. For example, the fact schema, which describes a person as an employee of an organization, consists of two roles:

- “person” (type “http://iks-project.eu/ont/person”) and

- “organization” (type “http://iks-project.eu/ont/organization”).

The list of roles must correspond to the list of defined roles in the fact schema. The values are URIs. Entityhub is used to resolve references to the entities that are identified by their URIs. When querying for the existing facts, Fact Store defines a simple JSON-LD-based query language that requires the user to be informed about the URIs of the participating entities. JSON-LD-based language can be extended to query for combinations of facts, which allows for a simple reasoning mechanism.

4.1.8 Apache Stanbol CMS Adapter

This component acts as a bridge between CMSs and the Apache Stanbol. It interacts with CMSs through JCR and CMIS specifications. That way, any content repository that is compliant with JCR or CMIS specifications, can make use of CMS Adapter functionality. One of its main features is bidirectional mapping between RDF data and content repository structure. In other words, it can transform the content repository structure into an RDF format or populate the content repository based on an external RDF data. Furthermore, it lets users commit of content repository items, together with their properties and enhancements to the Apache Stanbol Contenthub.

4.1.9 VIE Widgets

VIE.js[31] (also known as Vienna IKS Editables) is a JavaScript library that implements decoupled CMSs and semantic interaction in web applications. VIE bridges Backbone.js and Semantic Web technologies. It also enables easy interaction with RDFa annotated content and a connection with various semantic services, such as Apache Stanbol and DBpedia. VIE supports dealing with namespaces, relations between entities, content type system. It is also used as a basis for a wide variety of tools ranging from content annotators to full front-end editing interfaces and semantic browser extensions. In the following, we list several VIE widgets:

- Form generator: it generates Backbone Forms schemas;

[31] VIE webpage: http://viejs.org/

- Autocomplete: it uses VIE.find service method to make autocomplete suggestions. In addition, VIE.find method can query different backend and frontend data sources;
- Create.js content editing (c.f. http://createjs.org/): It is a comprehensive web editing interface for CMSs, designed to provide a modern, HTML5-based environment for managing content. It can be adapted to work on almost any CMS backend;
- Image Search: This widget uses Flickr API for image search;
- Autotagger: It displays a list of found entities in a tag cloud;
- Annotate.js: It is semi-automatic annotation editor developed to support rich HTML editors.

4.1.2 Semantic User Interaction: VIE Editor

The VIE Editor is a result of interaction pattern analysis supporting web-applications development. In its first release, VIE targeted the development of decoupled CMSs based on semantic annotations of a webpage. The underlying principle of decoupling CMS is to encode knowledge about the content directly in the content, allowing users to know how to deal with different parts of the content. This gives search engines a deeper understanding about webpages . The second release of VIE extended its capabilities to ease the development of semantic interaction. The API now offers a DSL to handle different namespaces seamlessly, maintain ontological hierarchies (including fully-typed, multiple inheritances) and access semantic backend services such as:

- VIE.analyze(): It analyzes DOM elements depending on the registered engines (e.g., RDFaparsing, Apache Stanbol Enhancer, Zemanta) and returns an array of found entities.
- VIE.load(): It loads all properties for the given entity from external services into VIE.
- VIE.save(): It saves knowledge about an entity to a service. This service can be the entityhub of Apache Stanbol, but also the local storage of the browser.

- VIE.find(): It queries semantic services, e.g., all Persons whose names start with "Bar".

By default, VIE comes with the ontology, which is provided by http://schema.org. However, VIE is ontology-agnostic and allows to easily extend, remove or change the ontology.

4.2 Ambient Intelligence Components of IKS

One of use cases in IKS implements an intelligent, Ambient Intelligence bathroom [IKS-D4.1]. Ambient Intelligence (AmI) has been characterized in many different ways; for example, “AmI implies intelligence that is all around us” [MAMI06] or that is “a vision of future daily life […]that intelligent technology should disappear into our environment to bring humans an easy and entertaining life” [CRUT06]. The diverse definitions assembled by Cook et al. [COOK09] highlight AmI features such as context-aware, personalized, anticipatory, adaptive, transparent, ubiquitous, intelligent. The vision of AmI focuses similarly on the human needs as on the technology development.

Figure 12 elaborates logical architecture for the AmI use case that is designed and implemented to demonstrate content and knowledge features of the IKS CMS technology stack. All modules of logical architecture are split into several components that communicate directly or via broadcast messages.

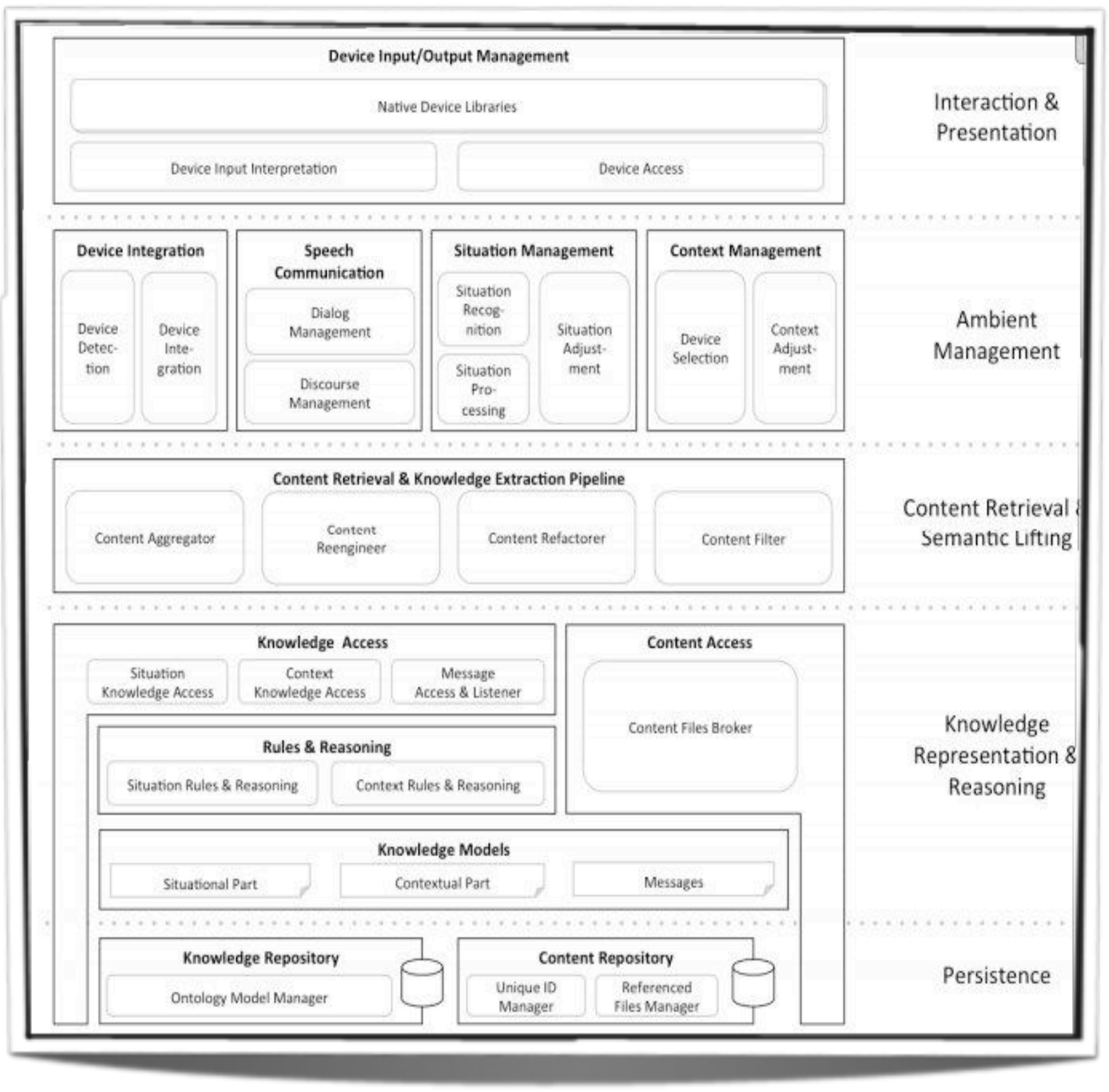


Figure 12: Component-specific elaboration of logical architecture of AmI system

In the following, we provide a description of AmI architecture modules and their components [IKS-D4.1].

4.2.1 Components of Device Input/Output Management

Device Input Interpretation Component is responsible for the interpretation of inputs sent from devices within the bathroom environment. Results are broadcasted to all major modules. The functionality of Device Input Interpretation Component encompasses three tasks:

(1) the component cares for the interpretation of device inputs (especially inputs from activity recognition devices);

(2) the component broadcasts interpreted device events to other modules, and

(3) the component redirects these inputs to the Context Management Module for the evaluation of inputs (i.e. in case that input of a device is too complex (e.g., audio streams)). Furthermore, it provides an interface that enables devices to communicate events.

Device Access Component is responsible for the status management of devices, within the AmI bathroom environment. It has ability to send content items to specific devices for presentation. The main task of this component is orchestration of the device status “busy” to avoid a simultaneous presentation of multiple contents on one device. In addition, it cares for the presentation of content items on devices that are currently integrated in the AmI environment. It also provides an interface for presenting content items on a specific preselected device.. After a successful content presentation, the component triggers the Device Input Interpretation Component to broadcast this information as an event.

4.2.2 Components of Device Integration

Device Detection Component is responsible for the detection of devices within the AmI bathroom environment. It continuously checks whether a device is already integrated into the environment. To complete the device detection cycle, it also detects whether devices already left the bathroom environment. The component further provides an interface to explicitly inform the Device Integration Component about devices, which are available in the AmI bathroom.

Device Integration Component is responsible for the integration of detected devices into the technical environment of AmI bathroom. One of the major functionalities of this component is retrieval of the semantic representation of the device that has to be integrated. Afterwards, this semantic representation of the device is integrated into the Knowledge Repository. The component also cares for the retrieval of the OSGi based native device library, integration of this library into the technical I/O environment, as well as update when the device is removed from the AmI bathroom.

4.2.3 Components of Context Management

Device Selection Component is responsible for the selection of appropriate presentation devices, for the presentation of specific content items. Based on the information about all devices available in the AmI bathroom, the component can filter these devices regarding information on how they fit to the content item (e.g., whether type of content item is supported by the output device) as well as the current physical context (e.g., privacy of a device). Semantic descriptions of device references are retrieved from the Situation Processing Component of Situation Management module that decides how to present a specific content item.

Context Adjustment Component retrieves events broadcasted by the Device Input Interpretation Component of Device Input/Output Management module. This leads to context adjustments (e.g., user movement). If such an event is received by the component, the context representation is updated in Knowledge Repository using the Context Knowledge Access Component of Knowledge Access module. If this event led to a context change, this is communicated to the Situation Adjustment Component of Situation Management module for further adjustments of the situation. In parallel, the Context Adjustment Component cares for management of the contextual part in Knowledge Repository.

4.2.4 Components of Situation Management

Situation Processing Component continuously checks whether some situation can be covered by a pro-active system action. This is possible if all requirements are fulfilled, which are necessary to perform the next step within the current situation.

Situation Recognition Component continuously compares the current situation state to determine if a change of situation type should be performed.

Situation Adjustment Component analyses context according to evaluation of changes of the current situation representation. This evaluation is triggered by general context changes that are communicated by the Context Adjustment Component of Context Management module to the Situation Adjustment Component. If a relevant change is evaluated, the current situation and the physical bathroom situation are adapted to the change that is recognized in the Context Adjustment Component.

4.2.5 Components of Speech Communication

Discourse Management Component is responsible for the interpretation of the context of discourse (i.e., information coming from already loaded interaction). The component receives an interpretation from native device libraries of the Device Input/ Output Management module and the speech recognizer. Speech recognizer checks a speech interpretation against the discourse history, in order to enrich and/or disambiguate interpretation. Once the discourse management task is performed, the interpretation of user intention is passed to the Dialog Management Component.

Dialog Management Component is responsible for the management of dialog interaction. After the interpretation of user intentions is passed from the Discourse Management Component, the dialog management component decides on the next steps which are required to perform user commands. If the intention is clearly specified, the component broadcasts the command or the content item requested by the user to the Situation Processing Component of Situation Management module or to Device Access Component of the Device Input/Output Management module.

4.2.6 Components of Content Retrieval & Knowledge Extraction Pipeline

Content Aggregator Component is responsible for registering and accessing external services, which can include web services, content repository services or CMSs, from which information can be retrieved via various protocols. The output of content aggregation is retrieved content from external sources. The retrieved content may appear in various formats.

Content Reengineer Component. After retrieving the content from external sources, an alignment with the contextual part of the knowledge representation is needed. The first step to achieve such an alignment is to transform the retrieved content into a common ontological representation. The Content Reengineer Component is responsible for such transformation process. Since RDF has become de facto method for conceptual description or information modeling, RDF is aslo used for representing knowledge in the IKS AmI system.

Content Refactorer Component performs the second step in the alignment process. Once external content is transformed to a common ontological representation, there is still a need for alignment to map external content concepts to AmI ODP concepts. To perform such a mapping, the content refactorer requires access to the contextual part of the knowledge representation using the Context Knowledge Access Component of the Knowledge Access Module. Refactoring is based on refactoring patterns, which are RDF descriptions of an alignment between two ontologies. Content Refactorer Component interacts with the Context Knowledge Access Component of the Knowledge Access module by making generated knowledge available to other modules of the system.

Content Filter Component allows integration of the content aligned/reengineered by the Content Refactorer Component/Content Reengineer Component, according to the user preferences. A preference is expressed considering some characteristic owned by the content item. For example, the *UserContentPreferences* ontology expresses the relation between AmI users and content items in terms of user preferences about specific content item features, while *UserEventPreferences* ontology expresses the relation between AmI uses and external event items.

CHAPTER 5: Showcases

5.1 Showcasing Horizontal Applications of IKS

Horizontal applications of IKS are designed to cover many different kinds of CMS providers, or individual users, such as applications that address the need of the news and media industry (e.g. Nuxeo), enterprise publishing and search market (e.g. WordLift, Alkacon, Adobe), or AmI ubiquitous technology industry.

5.1.1 NUXEO Integration for the News Industry

Nuxeo is a generic CMS platform commonly used to build content-driven applications such as office document management, document versioning and lifecycle management, multimedia asset management, while supporting metadata based browsing of a collection of photos, videos and audio assets. To demonstrate the integration of the existing IKS components into such a platform, we demonstrate semi-automated text enrichment for the AFP (Agence France Press) News Agency. AFP journalists produce text and multimedia (photo, audio and video) news reports about events all around the globe. Those reports are aggregated in topical feeds with a structured format, such as NewsML format, as defined by the International Press Telecommunication Council (IPTC). In addition, the NewsML feed includes text body of the article and metadata about the context of the event, i.e. the headline, the event date, the reporting date, the main geographical location of the event, the list of persons, organization and places mentioned in the body, the topical categorization according to the IPTC subjects hierarchy.

AFP News Agency provides journalists and editors with the tools that increase the precision of writing and assist in filling the correct values for structured fields, as intuitively as possible. As the first step, the journalist writes the title and the main body of the article. Once the article is saved in Nuxeo’s document repository, it can be further edited by adding metadata, attaching files, comments, reviews, etc. By integrating the Nuxeo’s editing tools with the Apache Stanbol, the default user interface is upgraded with a widget called “Manage links”. This widget allows user to link articles to semantic entities from DBpedia knowledge base, which are additionally indexed via the Apache Stanbol Entityhub component to provide fast and reliable response in a situation of overloaded DBpedia knowledge base. In addition, manual linking of articles and semantic entities can be skipped by initiating automatic analysis of a document. That way, text of the article is added to a processing queue for automated text analysis and linking. Once the automatic analysis of a document is done, the document automatically displays the new results (Common metadata, State, People, Organizations, Places). The resulting annotations can be used in two ways: (1) to export a NewsML XML file that encapsulates the text body of the articles, authorship metadata and semantic annotations; and (2) to provide fast and effective topic navigation in the article database.

Nuxeo's work on integrating Apache Stanbol services for entity indexing, detection and automated linking is demonstrated online at: http://temis.demo.nuxeo.com (demo/demo). In addition, Nuxeo presented Apache Stanbol integration at the SemWeb.Pro 2012 conference that was held in May in Paris:[32] as well as at the IKS Workshop in June 2012 in Salzburg[33]:

5.1.2 Adobe Integration with Stanbol Contenthub

Adobe integration of Apache Stanbol is based on Sling-Stanbol integration component. Apache Sling is a web framework based on the Java Content Repository (JCR), which is used to store and manage content. Sling applications use either scripts or Java servlets, which are selected based on simple name conventions, to process HTTP requests in a RESTful way. Both Sling and Apache Stanbol run on the Java Virtual Machine (JVM) and both adhere to the OSGi standards for modularization. In addition, Sling-Stanbol integrates IKS-related components such as annotateJS and VIE, that additionally enable user interaction with the content and its enhancement.

Sling-Stanbol demonstration video is given at: http://vimeo.com/31509786

---

[32] http://www.semweb.pro/conference/semwebpro2012

[33] http://vimeo.com/45633053

In principle, running the launcher of the Sling-Stanbol deploys its HTML interfaces under the local server. Users can connect to the Sling' WebDav server by filling in the "Connect to Server" window. After processing newly added file, it is sent to WebDav end-point and automatically enhanced via Apache Stanbol' Enhancement Engine. Users can browse the submitted files, see annotations, and edit the document.

Sling-Stanbol's Content observer component retrieves newly submitted files and submits them further to Apache Stanbol Contenthub. After populating the Contenthub, keyword search or faceted search can be done over the stored content.

5.1.3 Alkacon – Integration of IKS technology into OpenCMS

Alkacon's motivation to integrate the components of Apache Stanbol within its own OpenCMS technology was threefold: (i) to support semantically annotated content for Search Engine Optimization (SEO); (ii) to enhance editing of OpenCMS with inline-editing capabilities via VIE; and (iii) to provide semantic content enrichment by using VIE and Apache Stanbol components.

Alkacon was driven by the Google Web Toolkit (GWT) and Java technologies during implementation of its OpenCMS technology. Alkacon implemented GWT wrapper around the Vie JavaScript library, which enables a Java/GWT developer to make full use of the VIE capabilities. They also implemented a content service that translates OpenCMS data types and OpenCMS contents into the VIE entities. Apart the VIE-GWT wrapper, Alkacon started a new project called Acacia Editor (Acacia Editor webpage: https://github.com/alkacon/acacia-editor). It is based on an CMS independent editor that is highly customizable and currently supports web form rendering, inline editing of RDF annotated HTML content, interface for content retrieval and persistence service. As WYSIWYG-component for inline-editing, Acacia uses an adjusted version of Hallo.js (Hallo.js webpage: http://hallojs.org/) that was also initially developed in the course of the IKS project. Editable content elements are tagged with RDFa annotations. Next to semantic and the SEO benefit these annotations making the client able to tell the server which pieces of content have been changed and to which resource a concrete pieces of content belongs. Alkacon live demo is online present at: http://iks.alkacon.com

5.1.4 WordLift – Integration into WordPress

WordLift is a WordPress plug-in that is based on Apache Stanbol. It is aimed to enrich user-created text (a blog post, article or web page) with HTML Microdata annotations, which follow the Schema.org vocabulary that is already adopted by the major search engines, such as Bing, Google, Yahoo! and Yandex.

WordLift relies on Apache Stanbol Enhancer in two ways:

(1) named-entity recognition in text and their linkage to entities in Linked Data is realized by using NLP and Linking Suggestions Enhancement Engines of the Apache Stanbol;

(2) generation of Schema.org-compliant annotations for recognized named-entities is realized via Refactor Engine, which is a post-processing enhancement engine. Refactor Engine uses the annotation graph which generate fully semantically harmonized text annotations for named-entities.

WordLift[34] reads web pages or blog posts, understands it and enriches it by querying the Semantic Web and by adding the most relevant information using HTML Microdata. All the information retrieved can be manually edited by the author of the post (or page) and uses a markup vocabulary that all major search providers (Google, Bing and Yahoo!) recognized.

Through a simple Plug-In all available contents will be instantly compliant with schema.org specifications for a better SEO. It is currently enhancing content entities related to people and places.

5.1.5 Ambient Intelligence Integration

This use case combines semantically enhanced content and knowledge management within an interactive content-enhanced bathroom. From end-user perspective, it provides the following services, as shown in Figure 14:

[34] WordLift webpage: http://wordpress.org/extend/plugins/wordlift/

- **Weather Information Service:** A service that provides weather information in the bathroom. *Interaction:* Distance sensor in front of the mirror triggers today's weather information to be displayed on the mirror.
- **Event Recommendation Service:** A service that recommends events (e.g., theatre play, concert, movie in the cinema, etc.) in the bathroom. *Interaction:* The distance sensor in front of the mirror triggers three events to be displayed in the mirror.
- **Ticket Order Service:** A service that allows you to order tickets for events. *Interaction:* (a) Distance sensor in front of the mirror triggers event recommendations to be displayed in the mirror, (b) a ticket can be request by touching an event, (c) a verification question is asked via the speakers and (d) the array microphone listens to the answer by the user.
- **Personalized Music Service:** A service that plays music from a music collection in the bathroom. *Interaction:* Distance sensor in front of the eScreen starts the playlist and wiping along the touch-sensitive interaction border stops the music again.
- **Personalized News College Service:** A service that provides a personalized news collage (e.g., a news collage that addresses your interests sport and politics, etc.) in the bathroom. *Interaction:* (a) User asks for today's news from within the bathroom; the request is captured by the microphone array (b) the news are then displayed via text or video depending on the location of the user on the eScreen, the Shower or the mirror.
- **Adaptive News Service:** A service that provides the same news as described above but in different ways (e.g., via audio or via text and images) depending on the location of the user in the bathroom. *Interaction:* Distance sensors in front of the mirror, eScreen or Shower trigger the form (text or video) of the news such that the user can "take" them from one location to another within the bathroom.

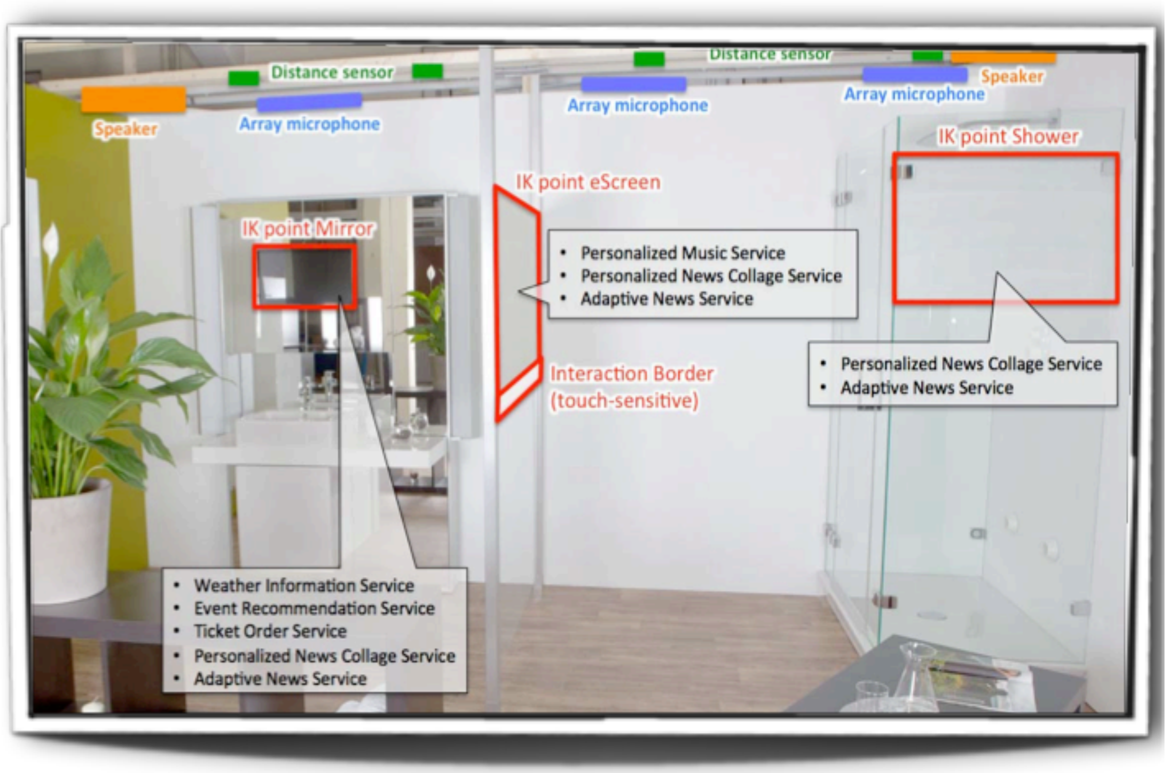


Figure 13 - Spatial placement of the six information and communication services. Note: IK point stands for Interactive Knowledge

From the technical perspective, the interactive AMI bathroom prototype is an instance of a modular Ubiquitous Information System (UIS), which consists of several loosely coupled, exchangeable modules. The main modules involved in the AmI case are as follows:

- **Knowledge Repository Module** (KRM). It manages storage and orchestration of knowledge representations and content items in the system. Every content item is referenced by URI, and all changes are communicated via semantically formatted messages. KRM cares not only about the message propagation between the modules. It also provides several utilities to simplify the work with the knowledge representations and content items. It provides rule-based reasoning capabilities and semantic listeners to enable access on the contextual and situational part of the managed knowledge representations.

- **Device Management Module** (DMM). All devices from a physical environment provide their semantic device metadata, which describes their functionalities. In this way, devices can be integrated and removed from the environment at runtime. Other modules can query devices which are currently available and use those devices for the presentation of contents.
- **Context & Situation Management Modules** (CSMM). It is responsible for a continuous update of contextual parts of the knowledge representation based on changes in the environment. The AmI ODPs (Ontology Design Patterns) are semantic representations of all concepts that are involved in the bathroom situation, e.g. the user and his preferences, content items like weather information, device descriptions like presentation devices, sensors or lights in the environment. The module continuously adapts semantic representation of the context to the current state in the environment. It also provides the capability to store content items of different forms retrieved from the Semantic Content Extractor Module.
- **Situation Management Module**. It uses the contextual part prepared by the Context Management Module in combination with the situational parts, i.e. semantic descriptions of situation patterns described as AmI Pre-Artifacts. This module searches for situation descriptions from the situational part of the knowledge representation that fit to the current situation and react accordingly. Hereby, a continuous analysis of the contextual representation is conducted. This process is managed by two components: (1) Situation Recognition & Processing and (2) Situation Adjustment. The first component addresses the recognition, processing and broadcasting of situational changes. Since situation recognition and processing tasks are highly interconnected, these two conceptual issues are realized in one component. By contrast, the Situation Adjustment component cares for the adjustment of the situation based on contextual changes in the bathroom environment, e.g., in case the user moves to another location.

- **Speech Communication Module**. It is responsible for speech interpretation/ generation, and discourse/ dialogue management. It recognizes spoken user input and interprets the user requests. As a result, hypotheses are made and checked against the situational context to identify the expected tasks and broadcast them to the system. After performing expected tasks, a multimodal presentation is produced and broadcasted to the Device Access Component.
- **Semantic Content Extractor Module** (SCEM). It is responsible for collecting external content from different sources and aligning this content with the AmI System. SCEM is composed of four subcomponents, such as: (1) Content Aggregator, (2) Content Reengineer, (3) Content Refactorer and (4) Content Filterer. These subcomponents work through the Content Retrieval & Knowledge Extraction Pipeline. The Content Aggregator obtains content either in XML or RDF format. If the content is in XML format, it is transformed into RDF format. Once the content is in RDF format, the Content Refactorer further transforms it into another RDF which is processable by the AmI System.

SCEM transforms content into a standard representation that is managable by the AmI System. It gathers content from NYTimes, BBC, WeatherBug, Google Calendar, Google Movies, Eventful, and Eventim. While serving the content, SCEM considers user preferences. For instance, it does not show social events that would overlap with existing calendar entries of users.

Video clip demonstrating AmI applications of IKS is online[35].

5.2 Showcasing Vertical Applications of IKS

In contrasts with an horizontal application, a vertical application addresses requirements of a single market, i.e. software that helps doctors manage patient records, insurance billing, etc. IKS vertical applications encompass a tourist use case (Pisano),

[35] http://www.youtube.com/watch?v=xIFX0Wjx3tc

5.2.1 Pisano Touristic Applications

Pisano integrated semantic technologies based on Apache Stanbol into its *packagemaster* touristic system. Semantic functionality should help product managers to analyse hotel informations and tour descriptions, and better retrieve the information. The objective of *packagemaster* system is to automatically generate additional information that better describe tour attributes and improve the quality of tour content. Also, an important goal is to reduce the time spent on data management.

The Product Editor (Webpage of Pisano's Product Editor: http://www.iks-pisano.de/) of the Pisano's packagemaster system is used for creating and maintaining all the information about a tour. This also includes descriptions that are provided in text form, which can be maintained by using the VIE widget (annotate.js).

For the content manager of a tour operator, who is not meant to or does not wish to maintain complex content, the website provides the option of signing in and editing defined text sections with the VIE widget (annotate.js) and adding additional information to them.

This functionality is ideal for making quick and simple adjustments without having to access the maintenance system. The ability to integrate additional data into the content through the IKS enhancer makes work much easier for the content manager.

For more details on the eTourism application see Pisano's websites[36].

5.2.2 Polymedia Publishing Applications

In the first year of the IKS project, Polymedia designed and developed a demonstrator that was based on their existing CMS. The demonstrator was aimed to provide an environment for testing the impacts of changes and integrating new components into an CMS-based editor. A scenario foresees a publisher who's writing an article about a considered subject. For that sake, Polymedia replaced the existing proprietary editor with the Hallo editor, which allows for creating annotations within

[36] http://www.iks-pisano.de, http://iks.cic.de/

textual documents. The following Figure 14 shows the automatic content enrichment via Hallo editor.

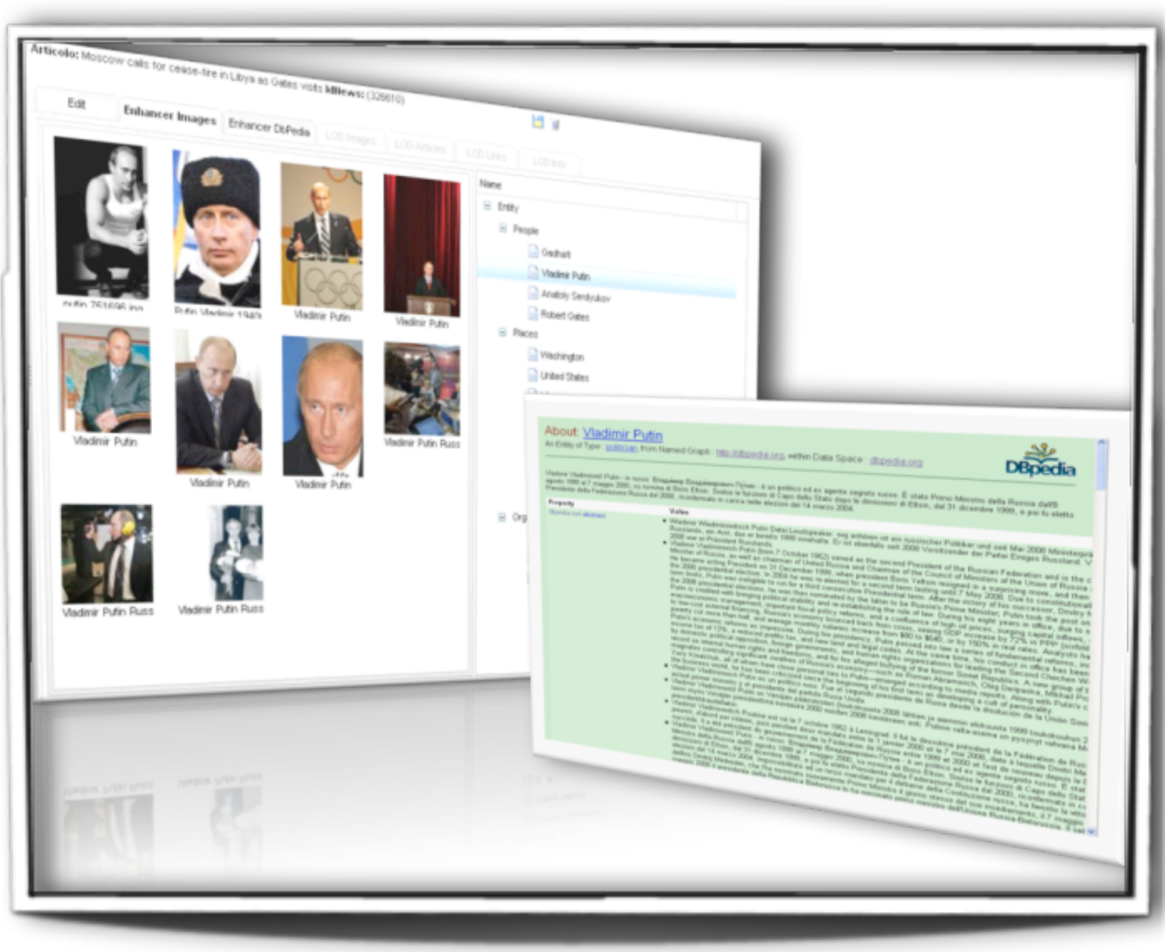
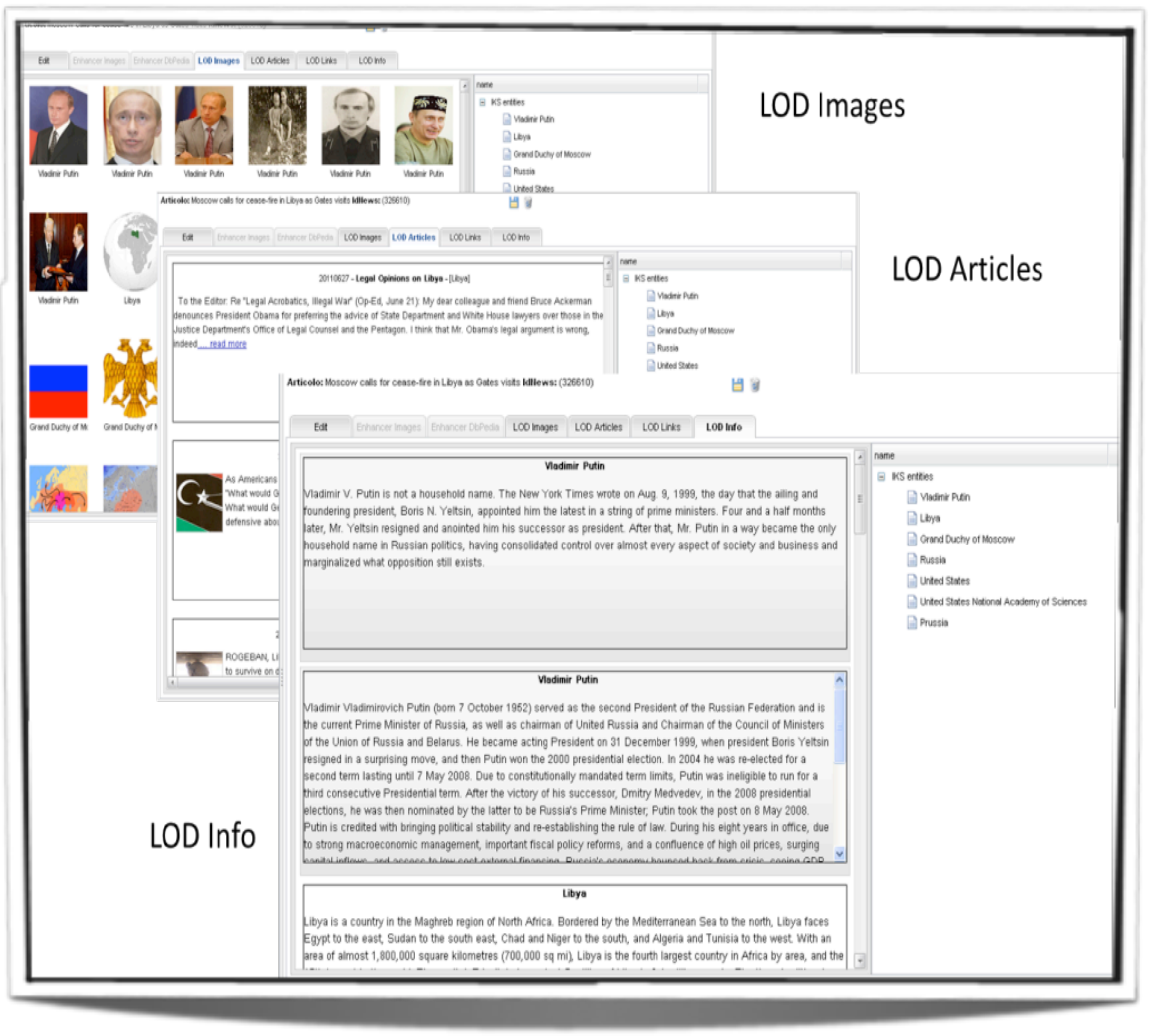


Figure 14: Polymedia Site Publishing

The following enrichment involves additional analysis based on LOD (Linked Open Data) (Figure 15). It also uses the Apache Stanbol Enhancer for extracting concepts, images and DBpedia links. The textual content and its generated metadata are then saved within the Polymedia CMS database for further processing and querying.



Articolo: Moscow calls for cease-fire in Libya as Gates visits **Idllews:** (326610)

20110627 - **Legal Opinions on Libya** - [Libya]

To the Editor: Re "Legal Acrobatics, Illegal War" (Op-Ed, June 21): My dear colleague and friend Bruce Ackerman denounces President Obama for preferring the advice of State Department and White House lawyers over those in the Justice Department's Office of Legal Counsel and the Pentagon. I think that Mr. Obama's legal argument is wrong, indeed ... read more

As Americans
"What would G
What would G
defensive abo

ROGEBAN, Li
to survive on d

Articolo: Moscow calls for cease-fire in Libya as Gates visits **Idllews:** (326610)

Vladimir Putin

Vladimir V. Putin is not a household name. The New York Times wrote on Aug. 9, 1999, the day that the ailing and foundering president, Boris N. Yeltsin, appointed him the latest in a string of prime ministers. Four and a half months later, Mr. Yeltsin resigned and anointed him his successor as president. After that, Mr. Putin in a way became the only household name in Russian politics, having consolidated control over almost every aspect of society and business and marginalized what opposition still exists.

Vladimir Putin

Vladimir Vladimirovich Putin (born 7 October 1952) served as the second President of the Russian Federation and is the current Prime Minister of Russia, as well as chairman of United Russia and Chairman of the Council of Ministers of the Union of Russia and Belarus. He became acting President on 31 December 1999, when president Boris Yeltsin resigned in a surprising move, and then Putin won the 2000 presidential election. In 2004 he was re-elected for a second term lasting until 7 May 2008. Due to constitutionally mandated term limits, Putin was ineligible to run for a third consecutive Presidential term. After the victory of his successor, Dmitry Medvedev, in the 2008 presidential elections, he was then nominated by the latter to be Russia's Prime Minister; Putin took the post on 8 May 2008. Putin is credited with bringing political stability and re-establishing the rule of law. During his eight years in office, due to strong macroeconomic management, important fiscal policy reforms, and a confluence of high oil prices, surging capital inflows, and access to low-cost external financing, Russia's economy bounced back from crisis, seeing GDP

Libya

Libya is a country in the Maghreb region of North Africa. Bordered by the Mediterranean Sea to the north, Libya faces Egypt to the east, Sudan to the south east, Chad and Niger to the south, and Algeria and Tunisia to the west. With an area of almost 1,800,000 square kilometres (700,000 sq mi), Libya is the fourth largest country in Africa by area, and the

Figure 15: Polymedia Site Publishing with LOD enrichment

Business validation of a new Polymedia demonstrator tool assumed creation of a specific multimedia use case. That use case foresees two actors such as *the publisher*, who's using the demonstrator tool to prepare the chosen video by adding tags and *the user*, consuming the video via a semantic player, that is capable of reproducing the multimedia content while interpreting the tags in real-time. The demonstrator tool leverages on the following components developed in IKS: Semantic Video Annotation Services (Editor + Player) and VIE Image widget (Player). In other words, the idea behind the demonstrator tool is to enable the semantic video annotations through the Polymedia CMS Video Editor (which is included in KIT Cosmos) (Figure 16) and the video playback harnessing the annotated video to provide an enriched consumption experience (i.e.; through VIE widgets) for the end-user.

The scenario is centred on the editorial office of a website dealing with cinema, using a video annotation software to add semantic tags to the various stages of the considered movie, while storing the generated semantic metadata in an internal repository. The Polymedia CMS Video Editor allows the content producer (e.g.; the journalist) to choose a video content and edit it before publishing, moving the timeline at the desired point, adding start/end markings, cutting and pasting video segments with single-frame precision, etc.

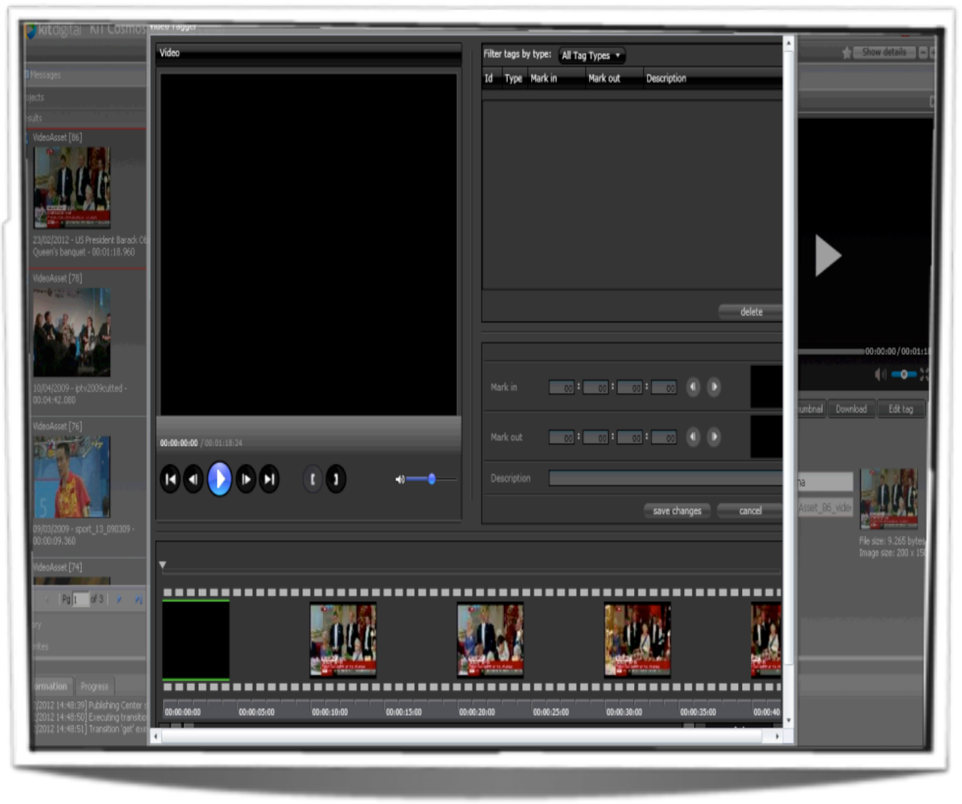


Figure 16: Defining a video segment within the Polymedia CMS Video Editor

The demonstrator tools is extended by allowing addition of semantic tags to a specific video segment: for example, the publisher can add semantic information about the scene and link it to a specific available plug-in. Every plug-in allows entering specific metadata. Once the annotation phase is completed, all resulting metadata are saved via IKS Semantic Video Annotation Services, which handle metadata into a format compatible with the type of data to be stored.

Concerning the Semantic Player tool (Figure 17), Polymedia initially developed a HTML5-based video player capable of playing back the video content published and annotated by the Polymedia/KIT Cosmos video editor. Such video player is compliant with the popcorn.js metadata format, thus interfacing video annotation services provided by IKS, and retrieving both the video reference and the metadata associated with it.

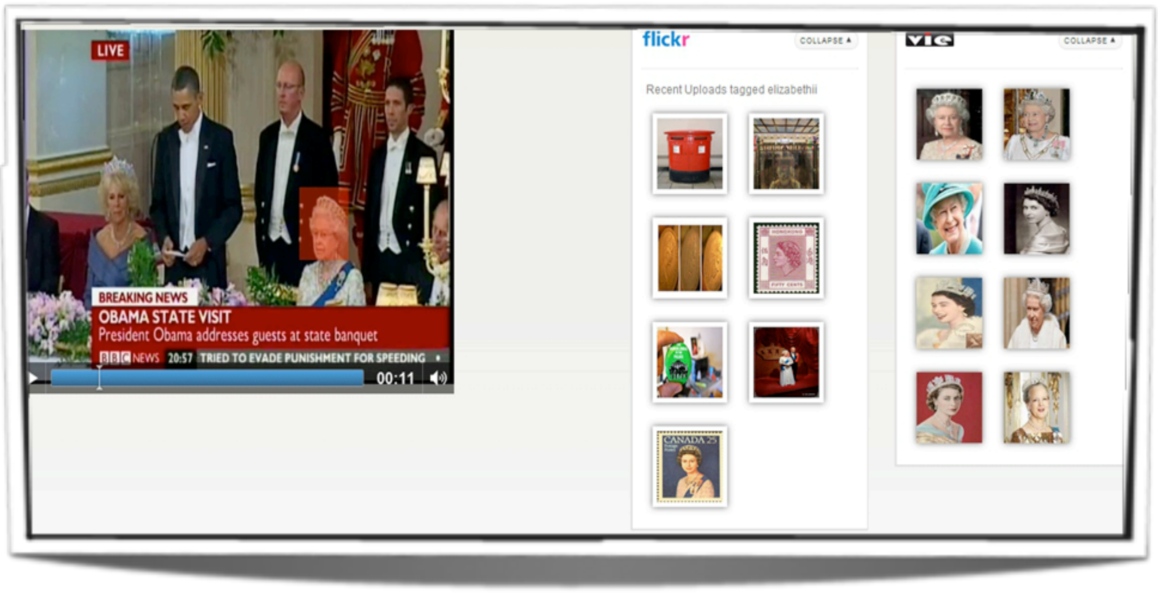
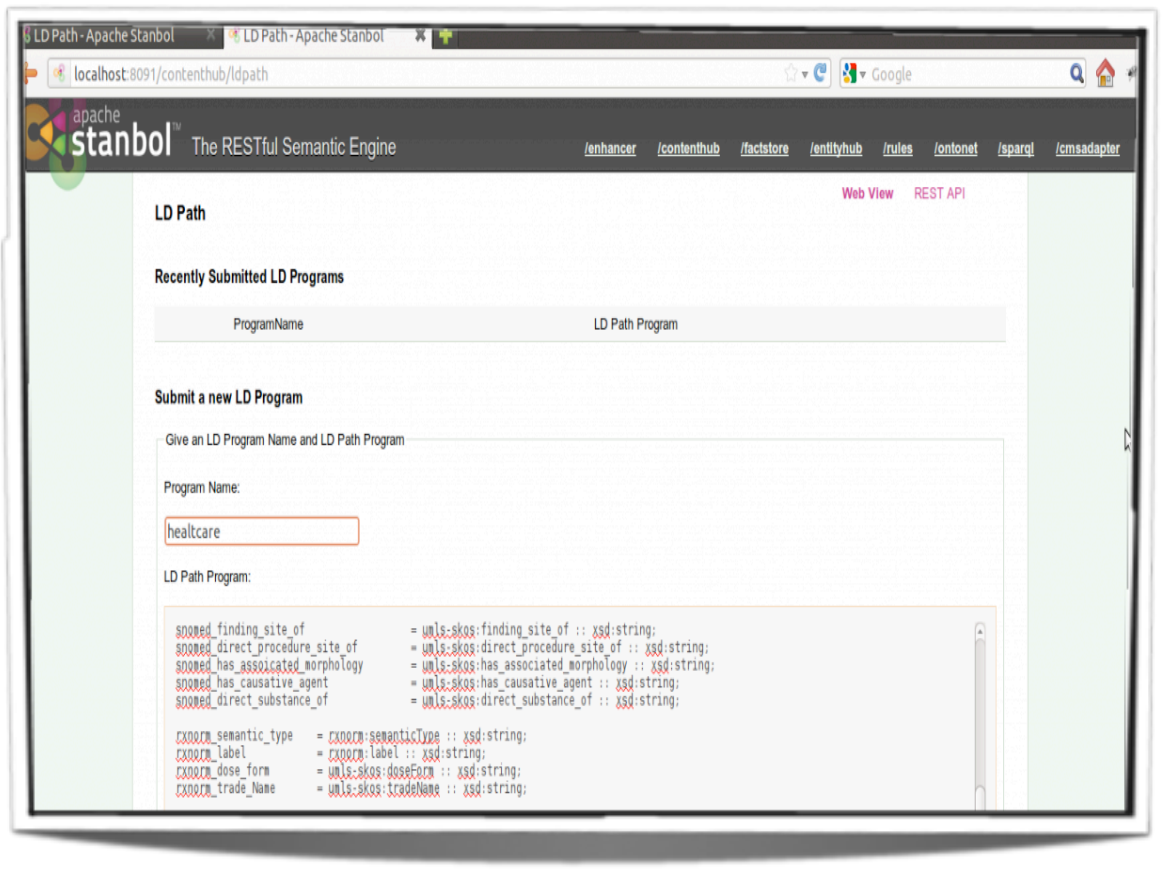

LIVE
BREAKING NEWS
OBAMA STATE VISIT
President Obama addresses guests at state banquet
BBC NEWS 20:57 TRIED TO EVADE PUNISHMENT FOR SPEEDING
00:11
flickr
COLLAPSE
Recent Uploads tagged elizabethii
COLLAPSE

Figure 17: A Semantic Player

5.2.3 Semantic Management of Health Related Data with Apache Stanbol

This showcase is based on health datasets from the LOD cloud, such as SNOMED/CT, RxNORM and ART (Adverse Reaction Terminology). After populating the CMS with health related documents, the documents are indexed via Apache Stanbol Contenthub, by creating an index in LDPath language. LDPath language is a query language for experimenting with the Linked Data Cloud. Figure 19 shows LDPath program submission interface.

After creating the initial index, the documents from the Adobe's CRX CMS are submitted to Apache Stanbol Contenthub via the CMS Adapter component. As a result, all the documents (Articles) are submitted to the newly created index. Once the content enhancement process is completed, Apache Stanbol Contenthub requests additional knowledge from Apache Stanbol Entityhub for each of the named entities that are recognized during the enhancement process. At the end of the indexing process, the new index is obtained embracing semantically meaningful information from the external RDF datasets. In addition, these information are used to provide semantic search functionalities for the documents. The next steps bring more refinement search, i.e. more restricted search on diseases, specific drugs and medications.



LD Path

Recently Submitted LD Programs

ProgramName	LD Path Program

Submit a new LD Program

Give an LD Program Name and LD Path Program

Program Name: healtcare

LD Path Program:

```
snomed_finding_site_of                  = umls-skos:finding_site_of :: xsd:string;
snomed_direct_procedure_site_of         = umls-skos:direct_procedure_site_of :: xsd:string;
snomed_has_assoicated_morphology        = umls-skos:has_associated_morphology :: xsd:string;
snomed_has_causative_agent              = umls-skos:has_causative_agent :: xsd:string;
snomed_direct_substance_of              = umls-skos:direct_substance_of :: xsd:string;

rxnorm_semantic_type    = rxnorm:semanticType :: xsd:string;
rxnorm_label            = rxnorm:label :: xsd:string;
rxnorm_dose_form        = umls-skos:doseForm :: xsd:string;
rxnorm_trade_Name       = umls-skos:tradeName :: xsd:string;
```

Figure 18: LDPath program submission interface

This showcase emphasizes several features, such as (i) the domain specific enhancement using the Apache Stanbol Enhancer component, (ii) the ability to create customized, use-case specific indexes, (iii) the ability to interact with JCR/CMIS compliant CMSs, to retrieve documents from these systems and use them in the domain specific enhancement and indexing features.

CHAPTER 6: Beyond CMS with semantic extensions ...

We are coming to the end of our excursion into semantic content management and it is legitimate to ask whether the technologies developed in IKS are on a trajectory that will revolutionize knowledge and content management; whether IKS was a credible but fated attempt to rescue a research strand that has run its course; or whether IKS has given the pragmatist in web engineering a useful set of tools with which modern web-scale applications can be built. In various research and technology forums, such questions are hotly debated and anybody with an interest in the topic should have the chance to assess the spectrum of opinions and lines of argument around semantic web technology and as a conclusion, about the role IKS technology is playing. We take the roles of the ardent supporter, the devil's advocate and the pragmatic software engineer to shed some light on the issues.

6.1 The Roaring Success of Semantic Web Technologies

In the previous chapters, we have already given a number of examples that illustrate how well the WWW has been served and enriched by semantic technologies. No other community has done more to promote RDF and this is now yielding benefits for the open government and linked data movements. Google and the other proponents of schema.org, who had first opted for microdata exclusively, have extended their range to include RDFa lite[37]. The GoodRelations ontology is being integrated into schema.org and is extending semantics potentially to every business on the planet[38], as long as it is connected to the Web.

With the development of the Web Ontology Language OWL, academics made it possible to open the arcane world of description logics to the WWW. Over the past 10 years, they made it possible to interoperate between relational databases, RDF and

[37] http://blog.schema.org/2012/06/semtech-rdfa-microdata-and-more.html

[38] http://blog.schema.org/2012/11/good-relations-and-schemaorg.html

OWL based reasoning engines and they developed SPARQL, a database query language that reaches across the different formalisms. Even the well-established world of relational databases has been challenged by RDF-based “triple stores” that combine traditional data storage with reasoning capabilities formerly known from expert systems, and all this is available over the Web and can be used very easily, with HTML5 enabled web browsers.

If we add to the technical and theoretical achievements, the concrete usage of RDF and some reasoning facilities, in large scale applications such as the BBC’s Olympic and football World Cup coverage then the case in favour of semantic technologies is evident and needs no further justification.

6.2 The Dismal Failure of Semantic Web Research

Twelve years after the famous article by Tim Berners-Lee, James Hendler and Ora Lassila in Scientific American, we can safely put the semantic web to REST (the pun with the upper-case characters is intended). What is left is *RDF* as a vague but useful data identification language for the WWW, *schema.org* as the only ontology of any noticeable size, that also has some wider use, and we have *RESTful services* as the single useful API on the WWW, with which everything is done that agents were promised to do, but these agents never appeared, for whatever reasons.

The reasons for the failure are as follows – we first present them briefly and then explain the details:

(1) No reference architecture - The research agenda never included Web Engineering and architecturally, semantic web discourse never got beyond the semantic layer cake. As several papers identified starting approximately in 2005, this layer cake was often mistaken for a systems architecture.

(2) No specific intelligence - The term “semantic” was intended to express that a future WWW would exhibit some form of intelligence or understanding towards the user, as opposed to just being a fast growing document store that would soon be too large to be indexed efficiently, for information

retrieval. The truth of the matter is that it is precisely the indexing and document retrieval technology, improved by large-scale statistics-based indexing that has kept the Web operational and has made Google one of the largest companies, world-wide. The technologies that are claimed successes of the "semantic" movement were known beforehand or have been adopted by the semantic web proponents in order to make their systems at least do something useful.

(3) No superior modeling – Undoubtedly, there has been impressive work on ontologies over the past 20 years, ranging from medical ontologies to general "upper ontologies" such as DOLCE and BFO, but applications that use these sophisticated ontologies have remained a rare sight, and the machineries are usually best handled by the inventors themselves. The more complex a knowledge model gets, the smaller the community that can do useful things with it. The vast majority of knowledge models however, is constituted of simple data schemas representing useful knowledge such as the famous "friends of a friend" (FOAF) or the "description of a project" (DOAP).

(4) No coherent development framework – You can use Protégé to develop ontologies or RDF schemas. You must use a triple store to manage RDF-instances. You need an OWL reasoner to compute OWL statements. You need an extended OWL reasoner to compute SWRL or RIF rule statements. You need a completely different infrastructure if you want to use semantic web services. And when you got all these glued together, you are still not connected with your Drupal, Typo3 or other CMS! And neither are you compatible with OODBMS or RDBMS.

(5) No convincing use cases – there are use cases of semantic web technology and several of them are successes. But do these suffice to give the technology a green light for broad adoption by smaller technology providers? Here, the argument gets into "shades of grey" because some cases that are hailed as semantic successes could also be interpreted as

"it would also work with conventional technology" and some cases could even be interpreted as "only the conventional elements make it work, the semantic elements are not actually the success factors".

(6) DARPA Challenges are missing for semantic technologies – The field of semantic technologies still defines its own challenges and these are often of the sort: "Look, we can translate between your (semantic web) tool and my semantic web tool".

6.2.1 No reference architecture

Around 1999, Tim Berners-Lee drew a diagram outlining the elements needed for the semantic web.

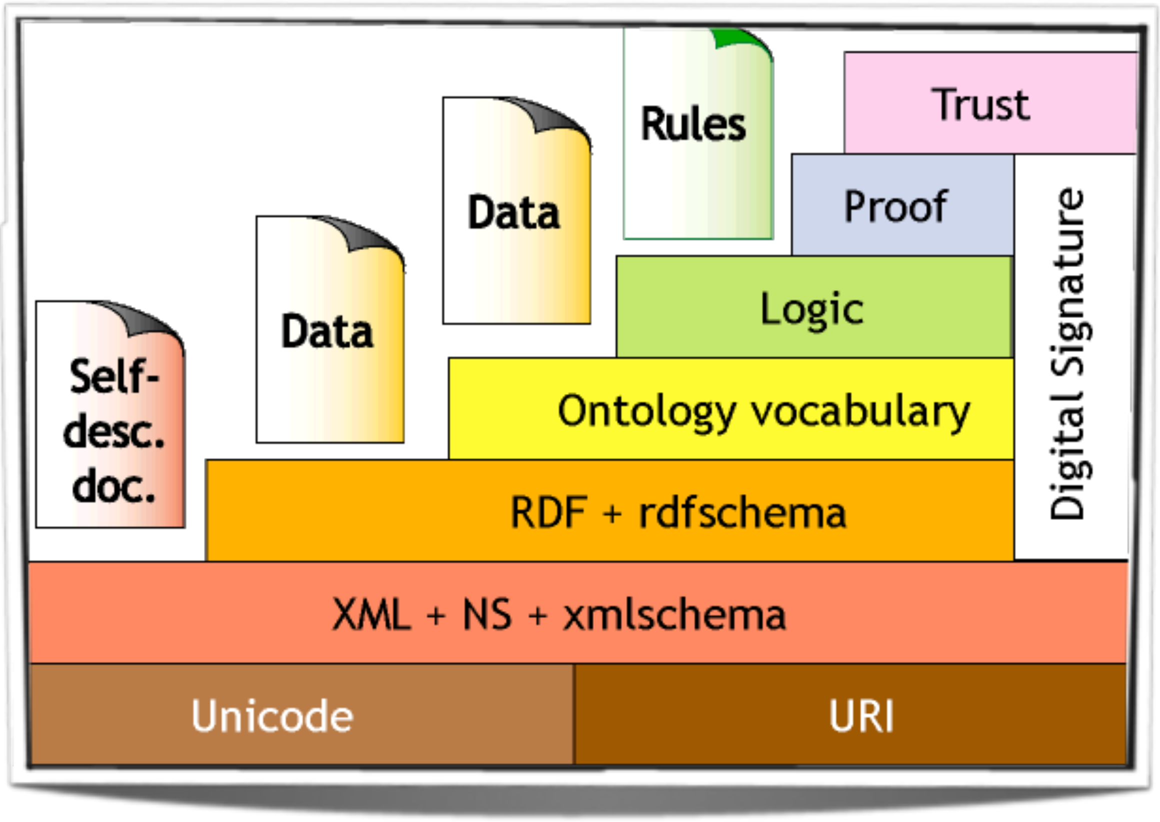


Figure 19: Semantic Layer Cake according to Tim Berners-Lee

However, what was originally meant to be an illustration of how technologies of the time could be used to create a real Stack, the layer cake illustration turned into an obstacle for independent thinking, because for too long, nobody dared to challenge the details, in fear of spoiling the idea.

A very entertaining talk in rhymes, by James Hendler called "*My take on the Semantic Web Layer cake*" (2009, http://www.youtube.com/watch?v=Pv9fpW6bhdo) illustrates well the complexity of the technologies *("this layer cake/ I must confess/ Really is a bloody mess/ / To many folks / it's causing stress!")*. The actual talk closes with the up-beat message that the community should be proud of what has been achieved, despite the large number of partially coherent technologies. From a systems engineering point of view, we would still emphasize that indeed, semantic web technologies are causing more stress than they yield benefits. Patel-Schneider proposed a revised "syntax architecture" [PATS05] to allow different syntaxes for various semantic web languages. While this is in principle, a solution to deal with different formal semantics and differing expressivity, it led the semantic web away from being a machinery for the mainstream developer and the average technology user. Horrocks et al [HPPH05] also analyzed the then emerging "stack" of semantic web languages. They were leaning more to the description logic approach and saw a danger in Datalog-type languages entering the language stack of the semantic web. They painted a picture of two competing stacks which they wanted to replace by one stack which had description logics (via OWL) firmly written into its core. Gerber et al. [GMB08] took the argument further by adding software engineering considerations, criticizing that the layer cake was mixing e.g. technologies and languages, and that despite everybody using the diagram, there was actually no paper formally defining the semantic web stack or cake. The authors then proposed a comprehensive layered architecture, which they claimed to have validated. The point for web engineering and content management is that the semantic web community was and still is, arguing over what kind of puritanism is better, while everybody else is building web-scale applications that do real jobs for real business! To make this point even clearer: what was termed an "architecture" discussion, above, was in fact a conflict over syntaxes and semantic scopes of data description languages. To this day, there is no coherent semantic web stack that starts with URIs and ends with a wellunderstood application layer. IKS was of course, also not able to fix these inherent problems, but is offering a reference architecture for semantically enabled CMS where at least the functionality is described down to API calls.

6.2.2 No specific intelligence

Assuming the above, language stack of the Semantic Web as a given, and accepting that each of the languages has one or more implementations that make it possible to write software applications with them, we should be able to work out specific implementation stacks with which programmed behavior can be achieved. If we follow the 2006 Semantic Web Stack (see figure) from bottom up, then we have:

URIs that denote resources in the virtual space. XML encodes any form of data including URIs. RDF is an XML-based, URI-enabled data specification language that describes multigraphs (graphs with different types of edges). RDF Schema is a data modeling language for RDF graphs. Then things become complicated: from 2006 on, the agreed-upon layer cake shows a remarkable departure from Tim Berners-Lee's original model: the original layer cake from 2001 presented ontology vocabularies as the step from the general languages to the concrete modeling level. We assume that the following layer of "Logic" in the 2001 model was meant to refer to the interpreters of specific ontologies, but the re-written version of 2006 replaced the "ontology vocabulary" layer by RDFS, OWL, RIF and SPARQL.

In other words, where originally, domain specific reasoning was supposed to be defined, we suddenly had yet more, generic knowledge representation notations and semantics, and the interpreting task was pushed to the next level up, called "unifying logic". This makes no sense at all because the idea of a stack is to become more specific as you go up its layers: so any "unifying logic" worth its salt would simply replace the four partial logics that it "unifies"! The fundamental criticism towards the 2006 "Stack" is: to build a reference model that can deal with a heterogeneous world is a good thing, but to build a reference model that is inherently heterogeneous just because of a conflict of opinion is simply a flawed model, because it has not even been hit by the modeling challenges of the outside world, yet, and it is already uncertain as to what it can express and how.

In 2007, Dan Brickley expressed his frustration with the architectural state of affairs like this (http://www.flickr.com/photos/danbri/428172848/):

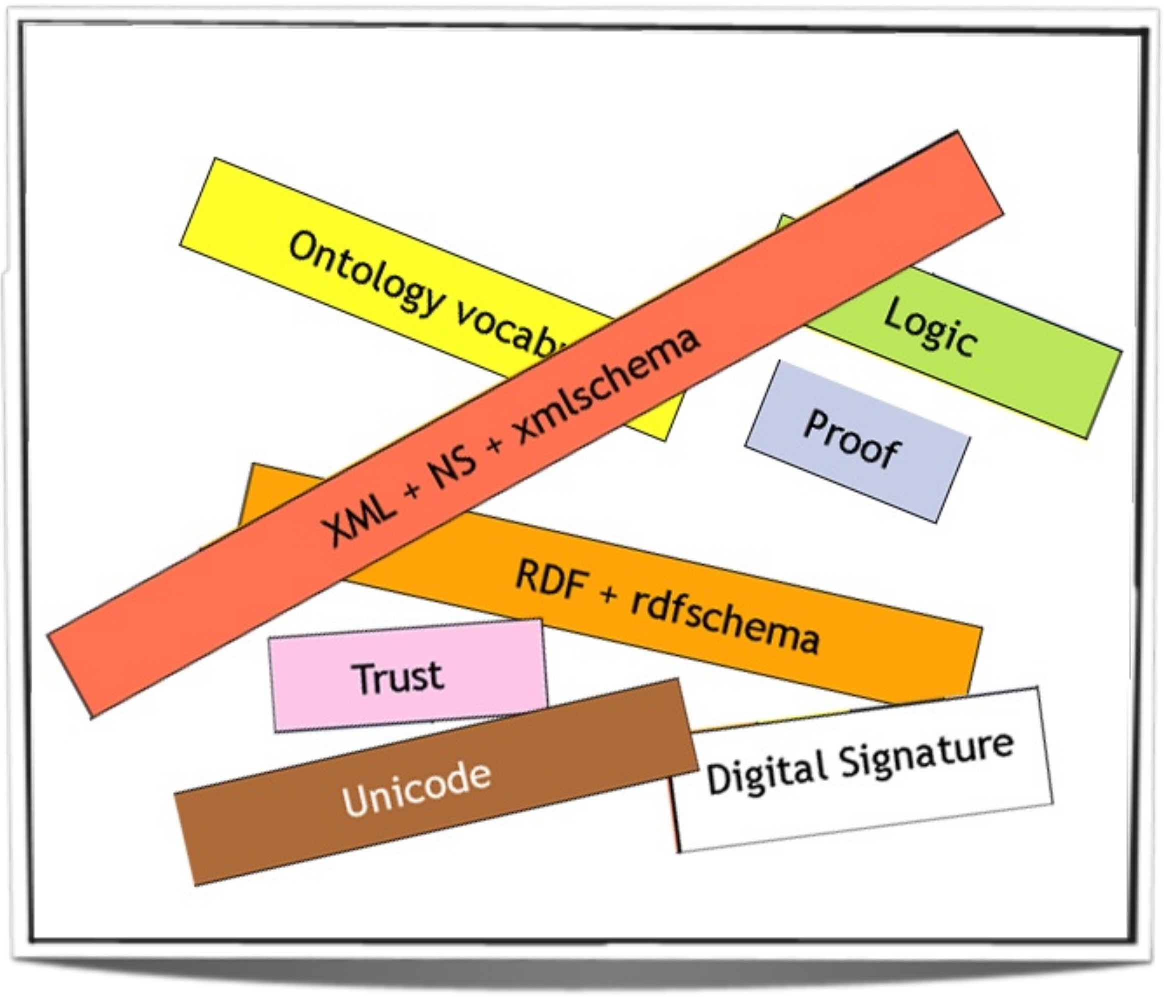


Figure 20: Dan Brickley's frustration with the architectural state of the semantic layers

In 2009, when James Hendler voiced his friendly criticism of the semantic web stack, and the "bloody mess" he referred to, looked like this (http://www.semanticfocus.com/blog/entry/title/introduction-to-the-semantic-web-vision-and-technologies-part-1-overview/):

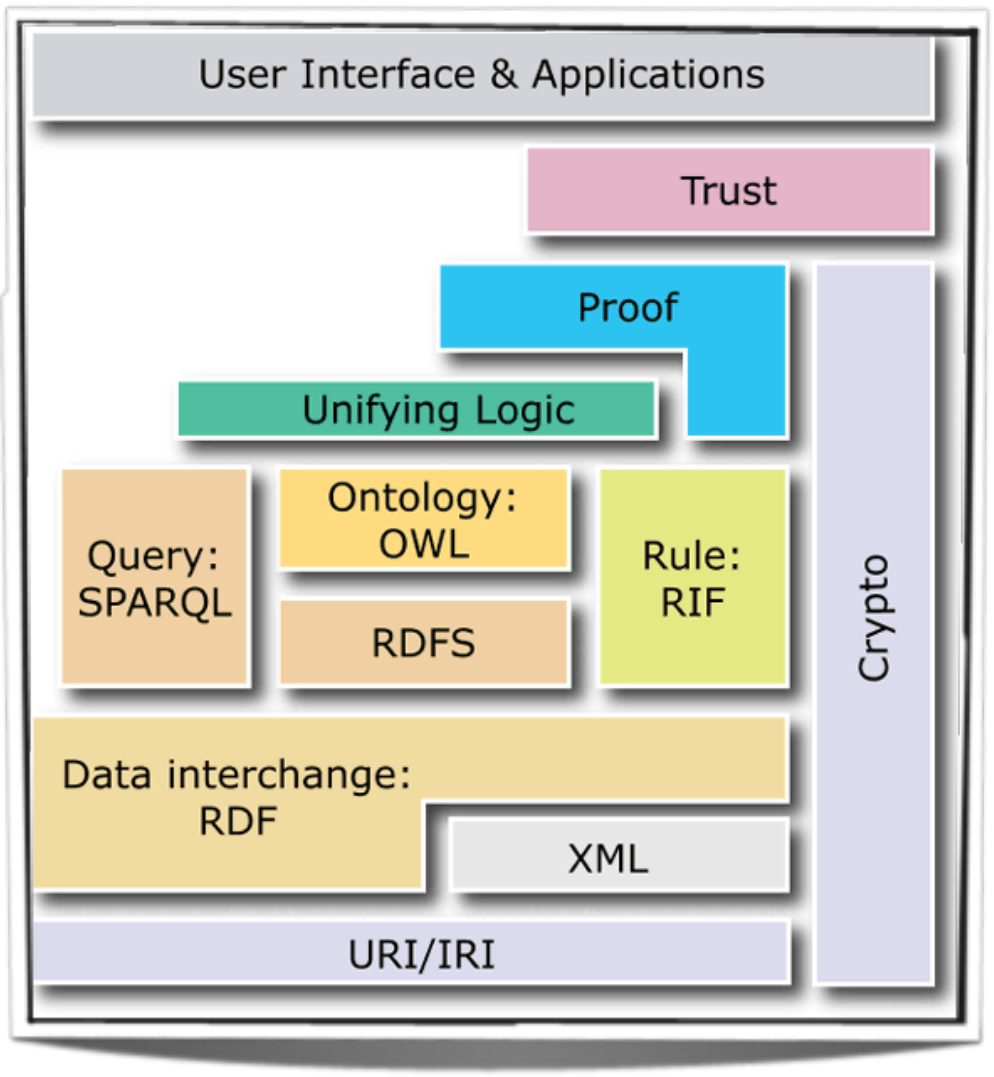


Figure 21: The 2009 consensus on the layers of the semantic web

There is wide agreement in the community that the unifying logic , proof and trust elements are still subject to research. If we take this seriously then it follows that "User Interface and Applications" still have to wait for some breakthroughs before we will see any methodologically sound Semantic Web applications. As the diagram also shows there is a further inherent conflict with the introduction of RIF. While SPARQL as a query language is in fact, offering complementarity to the data description languages of RDFS and OWL, RIF on the one hand, tries to remain an interchange format for different rule languages and semantics, but on the other hand, also needs (yet other!) implementations of such languages in order to be usable. The result is a quagmire of restrictions and incompatible modeling choices at the core of what the semantic web is supposed to support: machines that model understanding of human intention with respect to complex situations (i.e. any aspect of the world that can be supported by federated computing systems). We have given this section the heading “no intelligence”: the reason is that at the core, proponents of description logics have “won” the academic battle, but have lost the war when it comes to commercial applications and industrial uptake. In IKS, this is reflected in the hard work that the research group on reasoning had to put in, in order to make OWL-based machinery work together with serious content management applications and with RDF-based data repositories. One major problem was a lack of modularity at the beginning, on the part of the semantic web “engines” that turned out to be too monolithic for straightforward interoperation with existing CMS.

6.2.3 No superior modeling

It is often claimed by the proponents of the semantic web that building ontologies is somehow, a more coherent, complete and sound way of doing knowledge representation. At least in computing, we need more than knowledge *representation*. In most cases, computational modeling is the capture of some essential *behavior* and the attempt to program a machine to exhibit that essential behavior. To model any sort of behavior in a machine, we need three elements: defined components, defined interactions between components and some way of specifying the desired behavior as the result of components interacting in response to some stimulus. The original idea of the Semantic Web was still true to this idea: Ontologies should define the components (note that ontologies were seen as a means to an end), agents should embody the interactions and should be programmable for some purpose. Finally, RDF was seen as a sufficient coding standard to represent information on the web. Since agents were not academically fashionable at the time, they were conveniently forgotten by the semantic web research community. Similarly, since real ontologies about real domains require real understanding of both knowledge representation and of the target domains, young researchers preferred to invent new notations and stay clear of murky modeling issues in specific domains. In other words, the semantic web got stuck on the first rung: none of its specified languages goes anywhere beyond letting you define static components for arbitrary ontologies. It does not help you with the description of semantic “modules” (components and their interactions) and it does not offer any modeler-friendly tools to specify desired behavior. On the contrary, it forces the developer to cope with inherently inconsistent tools. Even at the level of just making information persistent, it forces developers to fight with tools whose “impedance mismatch” could also be described as “incompatibility”. Researchers like Pat Hayes and Ian Horrocks spent years, arguing over the formal semantics of RDF vs OWL or DAML. Similarly, we have now LD-Path vs SPARQL and most web-spanning queries start with good old information extraction techniques to first establish which data sources can be tackled with which querying tool. The Linked Data movement seems to have given the Semantic Web communities a way back into mainstream web engineering: small RDF-encoded ontologies for large datasets and simple forms of reasoning that can be achieved with a variety of means (even by clever indexing, using information retrieval techniques).

6.2.4 No coherent development frameworks

When there is a lack of coherent underlying models, it follows that there will be a lack of coherent development tools and frameworks. Nonetheless, the fact that there *are* commercial products available suggests that there is a core set of technologies for which entrepreneurs have seen market opportunities. Tools such as TopBraid™ provide an Eclipse-based environment for building ontologies using RDF, RDF schema and OWL. They also have OWL reasoning and usually SPARQL for querying RDF graphs / OWL ontologies. TopBraid™ offers four ways of handling rules: Jena Rules, using the Jena inference API, OWLIM, or their own implementation of SPIN (SPARQL Inference Notation). The use of the engines can be configured in the development environment. In order to deal with all notational eventualities, we can view the source code as RDF/XML, Turtle, or N3, and we can switch on or off, whether we want to see imported classes – in other words – whether we want to see the inheritance hierarchy of a given class. This makes for an impressively complex tool, but without further libraries, it still does not get us to deploy a working semantic Web application. Heitmann et al. [HCHD11] identify four challenges and at least three of these are "home-made" by the choices made by the proponents of semantic web technology: a) Mismatch of data models between components: graph-based RDF vs object oriented vs relational; b) distribution of application logic across components, and c) missing guidelines and patterns.

We can see – over the past two or three years - that research groups are beginning to tackle these issues with the development of Linked Data Servers, such as the currently incubating Apache Marmotta (http://marmotta.incubator.apache.org). It remains to be seen whether these tools give semantic web technologies a second chance or whether they just salvage a few useful elements that will then become part of mainstream web application building.

6.2.5 No convincing use cases

The majority of real-world applications that have been reported come either directly from research institutions or have been developed for validation purposes, by industrial companies in the course of large research projects. Even projects that can be attributed to a single industry player were usually done in one the organisation's research department and not by their IT department. The W3C maintains a list of use cases with 48 entries at the time of writing (January 2013): (http://www.w3.org/2001/sw/sweo/public/UseCases/)

We concede that use cases do exist and that there are applications particularly in data integration, that show potential. The use case descriptions at W3C are too superficial to conclude anything in terms of "how much semantics?" or "depth of reasoning" and the community has found it difficult to prove any unique selling proposition of semantic web technologies. The recent case of "IBM Watson" a software application that beat human contestants at the quiz-game "Jeopardy" uses – according to Chris Welty, one of its designers – some semantic web elements. However, Welty has pointed out that Watson also uses a number of other techniques that make up its overall performance, and that its performance would be impossible to achieve with just relying on Semantic Web technologies.

6.2.6 No real world challenges

From very early on, the semantic web communities detached themselves from serious domain-driven modeling efforts. At the beginning this was tolerable – for a period of time a new field should be granted to find its feet. But after a few years, the new field should emerge with convincing answers to some significant challenges. A real challenge is of the sort that DARPA set, for autonomous vehicles, in 2004:

The first competition of the DARPA Grand Challenge was held on March 13, 2004 in the Mojave Desert region of the United States, along a 150-mile (240 km) route that follows along the path of Interstate 15 from just before Barstow, California to just past the California–Nevada border in Primm. None of the robot vehicles finished the route. Carnegie Mellon University's Red Team and car Sandstorm (a converted Humvee) traveled the farthest distance, completing 11.78 km (7.32 mi) of the course before getting hung up on a rock after making a switchback turn. No winner was declared, and the cash prize was not given. Therefore, a second DARPA Grand Challenge event was scheduled for 2005.

(Source: http://en.wikipedia.org/wiki/DARPA_Grand_Challenge#History_and_background)

Only a year later, practically all contestants reached the finishing line and the challenge became a tightly fought race.

There have been several semantic web challenges over the years, but none so far, has been of the “Jeopardy/Watson” kind, or like the DARPA Grand Challenge.

This leaves us – when taking the role of the devil’s advocate – to conclude that “Semantic Web Technology” is a misnomer: Semantic Web has remained primarily a fringe field of ICT research, with a mixed agenda and with fragmentation of approaches that range from simple SKOS based thesauri encoded in RDF, to description logics-based reasoners interpreting one of at least four possible rule languages. And it leaves the Web-Developer bewildered at the lack of clarity, from a field that claims “declarative knowledge representation” as one of its unique value propositions.

For readers who wish to follow in-depth, critical views and occasional defenses of semantic web technologies, we recommend the archives of the ontolog-forum mailing list.

6.3 A Pragmatist's View on Semantic Technologies in Web Content Management

Having taken the roles of staunch supporter and devil's advocate for semantic web technologies in the previous sections, we shall now return to the line that has dominated work on the IKS project, namely, focusing on concrete improvements that can be taken up by developers of web-based content management applications. This – pragmatist's view – is reflected in the main contributions that IKS has been able to make, in the past four years. We will summarize the state of affairs achieved, by revisiting the major components:

- Web-based content editing (create.js)
- Management of editable objects (VIE)
- Apache Stanbol components
    - Enhancer
    - Entityhub
    - Contenthub
    - Ontology Manager
    - Rules
    - Reasoners
    - CMS Adapter
    - FactStore

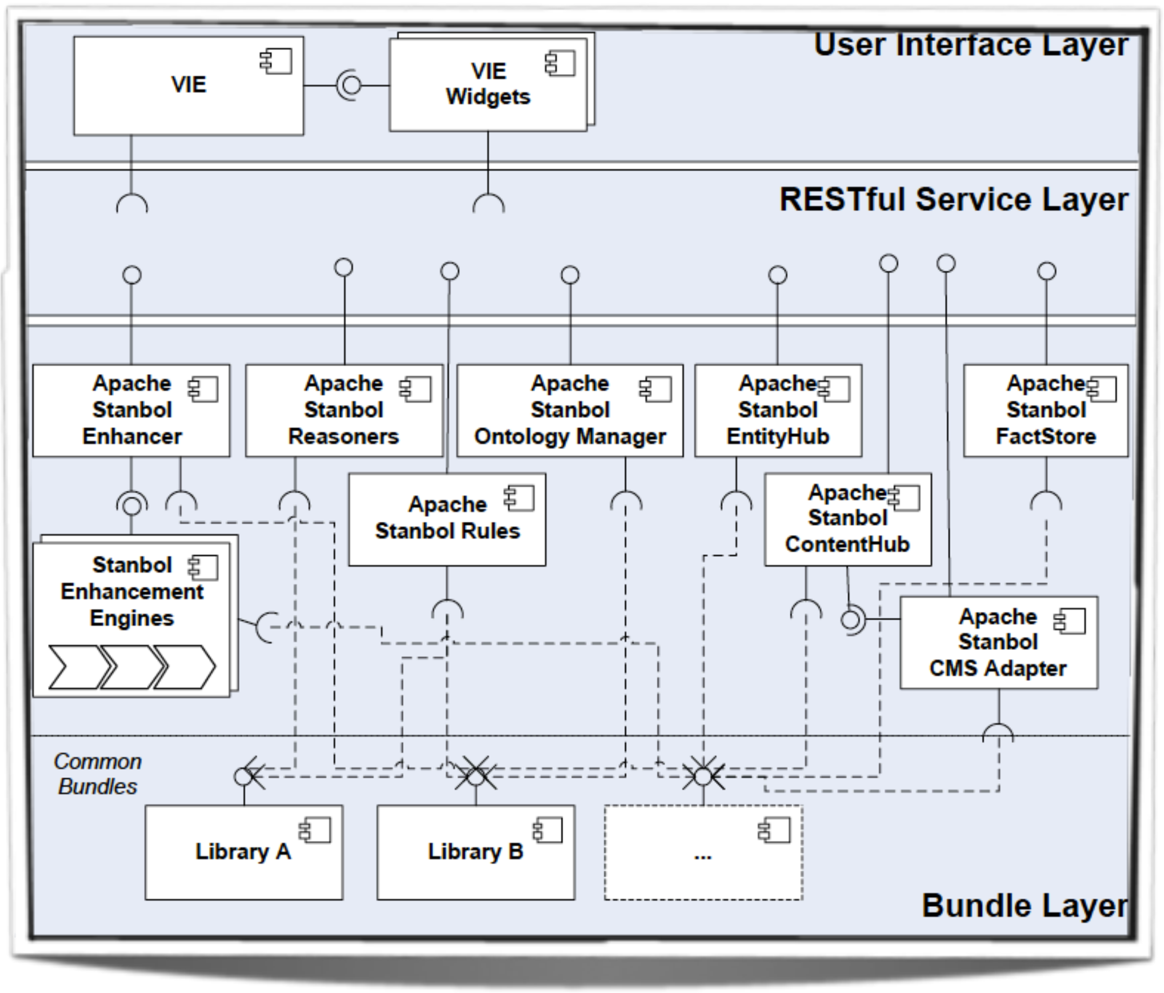


Figure 22: IKS layers and components (F. Christ et al. 2012)

6.3.1 Pragmatics of the User Interface – create.js and Vienna IKS Editables

In IKS we set out with a “grand vision” of AI-type, intelligent user interfaces that would turn an application into some smart agent that lets the user interact with “knowledge content” that would be increasingly available on the WWW. In the course of the first year, we had to accept that the reality of web based content management starts with issues that research has long declared as solved. One of these issues is the separation of form and content. It seems that every new generation of developers is faced with sufficiently new technologies so that the issue must be rediscovered or at least, re-interpreted. One such re-interpretation of federated information management was given by Henri Bergius, in the context of web-centered content management [BERG11]: he proposed to address the problem of web-frameworks that force developers to accept monolithic content management stacks. This led first to the development of a web-editing library (VIE) and later to a generalization (create.js) that offers an important connecting element between web-based application building and semantic web technology. This important practical issue of contemporary CMS would have been ignored if we had taken a purist's view insisting solely on tackling "new frontiers". However, there was a price to pay: the question how a complex knowledge domain can be represented and interacted with, via a user interface that is somehow "driven" by the ontology, was not answered by IKS. Instead, the question how to interact in a principled fashion, with content *and* with knowledge statements expressed in RDFa, has been answered! This has opened the door for semantic technologies to be more easily interfaced with content management while remaining modular on both sides.

6.3.2 Semantic Enhancement Engines

Big data stores are a very valuable asset as the history of Google teaches us. Big multilingual content stores are even more valuable, because the holder of the data can use large scale statistical methods to gain insights and learn from the data, as the examples of facebook or twitter teach us. The European Union is banking on machine translation [EUR13] in order to keep the costs of translation services low despite an increase in translations needed. These examples work well for the large organizations, because they are able to meet four preconditions:

a) they have a very large quantity of data available

b) the institutions/companies have extremely large computing resources

c) there are human experts in the loop that can kick-start machine learning

d) there are expert developers who can build a variety of information processing tools

In a typical SME setting, none of these four pre-conditions are met sufficiently well: the data is limited to what the end user organization has and needs for its business, the ICT resources are geared towards the primary business and humans are also focused on the primary business and have no time for anything that is considered “extra work”, and finally, developers in SMEs do not find the time to acquire deep information processing knowledge. This has resulted in a situation where SME technology providers have fallen far behind the quality standards expected by end users who are normally served by the machineries of the large organizations, from Google to Amazon, Facebook or ebay.

With its semantic enhancement engines, IKS has lowered the barriers for technology providing SMEs mostly with respect to d) by building a range of tools that make information processing tasks more feasible for SME Web CMS developers. This means that technology providers can create added value applications for their customers, more easily. Enhancement engines follow the UNIX principle of doing relatively small tasks (e.g. Named Entity Recognition) that can be combined to form “Enhancement Chains”.

6.3.3 Semantics meet Content – Entityhub and Contenthub

In order for a semantic system to work, we need some kind of knowledge base. In traditional content management, the best one can expect is a controlled vocabulary from which metadata is generated, manually or automatically. In IKS, we developed the ***Entityhub*** as the center of semantic data management. Content gets analyzed, indexed and the “entities” that we recognize, get stored in the entity hub. What’s more, the entity hub allows us to manage also external resources: a “managed site” is a resource that we analyze and replicate within the entity hub. Managed sites can also be connected to our internal entity hub via mappings to our own terms. Thinking federated, a “referenced site” is one which the entity hub is aware of, but which is only accessed by our system “as is”, i.e. there is no local replication and we accept the terms and availability of the external resource as a given.

These management schemes for semantically enhanced content have proven very useful for CMS technology providers because the schemes give them control over how a client wants to deal with internal and external resources, within an intranet information space.

The Stanbol ***Contenthub*** is an Apache Solr based document repository which enables storage of text-based documents and customizable semantic search facilities. The Contenthub exposes an efficient Java API together with the corresponding RESTful services. A document within Contenthub is referred as a "Content Item". A content item consists of metadata of the document in addition to the text-based content of the document. Contenthub has two main subcomponents, namely Store and Search. Store is responsible for persistent storage of content items while Search provides strong semantic search facilities over the content items.

6.3.4 Keeping your semantics in order – Ontology Manager

The Apache Stanbol ***Ontology Manager*** provides a controlled environment for managing ontologies, ontology networks and user sessions for semantic data modeled after them. It provides full access to ontologies stored into the Stanbol persistence layer. Managing an ontology network means that you can activate or deactivate parts of a complex model from time to time, so that your data can be viewed and classified under different "logical lenses".

Stanbol OntoNet implements the API section for managing OWL and OWL2 ontologies, in order to prepare them for consumption by reasoning services, refactorers, rule engines and the like. Ontology management in OntoNet is sparse and not connected: once loaded internally from their remote locations, ontologies live and are known within the realm they were loaded in. This allows loose-coupling and (de-)activation of ontologies in order to scale the data sets for reasoners to process and optimize them for efficiency.

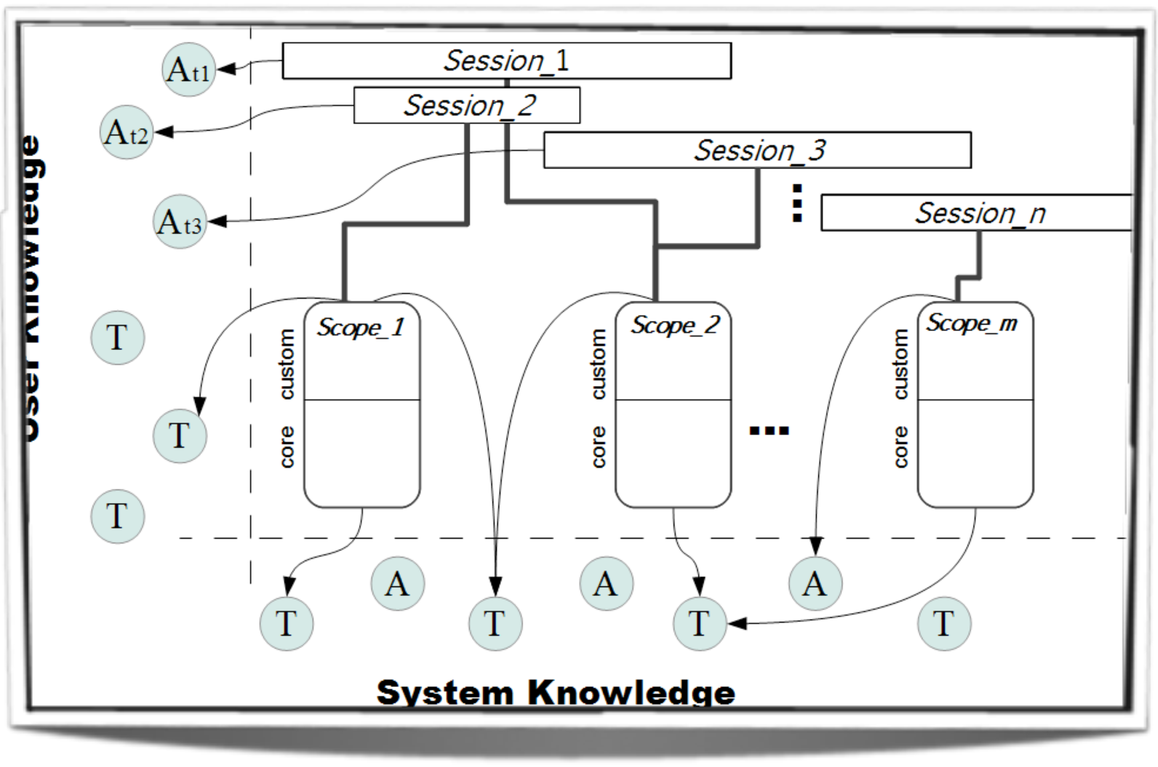


Figure 23: OntoNet setup with sessions, scopes and spaces.

OntoNet allows the construction and management of ontology networks, programmatically via its Java API or RESTful Web Services[39].

6.3.5 Rules and Reasoning

The Stanbol ***Reasoners*** component provides a set of services that take advantage of automatic inference engines. The module implements a common API for reasoning services, providing the possibility to plug in, different reasoners and configurations in parallel. Currently, the module includes OWLApi and Jena based abstract services, with concrete implementations for Jena RDFS, OWL, OWLMini and the HermiT reasoning service.

[39] http://stanbol.apache.org/docs/trunk/components/ontologymanager/ontonet/

The Reasoners module expose a REST endpoint with the following preloaded services:

- /rdfs, which is based on Jena RDFS reasoner and supports almost all of the RDFS entailments.
- /owl, a Jena reasoner configured to support OWL (with some limitations,)
- /owlmini, another Jena configuration that partially supports OWL

In addition a service which uses the HermiT reasoner to exploit the full OWL 2 specification is also available.

Each reasoner can be accessed with one of three tasks:

- check: to perform a consistency check. This service returns HTTP Status 200 if data is consistent, 204 otherwise (at the current state of implementation the service does not include an explanation about why the input is inconsistent)
- classify: to materialize all inferred rdf:type statements.
- enrich: to materialize all inferred statements.

6.3.6 Connecting with CMIS compliant systems – CMS Adapter

The ***CMS Adapter*** acts as a bridge between content management systems and Apache Stanbol. All components of Apache Stanbol also provide RESTful services which allow accessing them directly from outside. The CMS Adapter interacts with content management systems through JCR and CMIS specifications. In other words, any content repository compliant with JCR or CMIS specifications can make use of CMS Adapter functionalities. Currently, the CMS Adapter offers two main functionalities: "Bidirectional Mapping" and "Contenthub Feed".

Bidirectional Mapping provides two-way mappings between JCR/CMIS compliant content repositories and external RDF data. When using this feature it is possible to generate RDF data from a content repository or to populate a content repository based on the external RDF data.

The *Contenthub Feed* is realized by a two-step process involving the sequential execution of RDFBridge and RDFMapper services of CMS Adapter. Considering the update of the content repository based on external RDF data, in the first step the given raw RDF data is annotated with standard terms by RDFBridge. There are a few terms that are described in the *CMS Vocabulary* section. The RDFMapper processes the annotated RDF and updates the content repository accordingly. From the other direction, in the first step, the content repository structure is transformed into RDF annotated with the CMS Vocabulary terms by RDFMappers. In the second step RDFBridges add implementation specific annotations.

The bidirectional mapping feature makes it possible for content management systems, to exploit open linked data that is already available on the web. By mapping external RDF data, any existing content repository items can be updated or new ones can be created. The Contenthub Feed feature aims to manage content repository items within the Contenthub component of Apache Stanbol. The management process includes only two types of operations, submit and delete. Submission and deletion operations can be done based on the identifiers of paths of the content repository items. During the submission process, properties of content repository items are collected and they are stored along with the actual content. This enables the implementation of faceted search over the properties of items.

6.3.7 From Entities to Relations and Statements – FactStore

The FactStore manages relations between entities identified by their URIs. A relation between two or more entities is called a *fact*. The FactStore lets users store n-ary facts according to a user-defined fact schema. The FactStore only stores the relation and not the entities. It only uses references to entities by using the entities' URI. The entities themselves should be handled by another component, e.g. the Entityhub. A fact is defined by a fact schema which is defined over types of entities.

A fact schema can be defined between an arbitrary number of entities. In most cases a fact schema is defined between two or three entities. For example, the fact schema 'works-for' can be defined as a relation between entities of type 'Person' and 'Organization'. The Fact Store interface allows the creation of custom fact schemata and to store facts according to these custom schemata. The Fact Store provides a simple way to define and store facts. This component is meant to be used in scenarios where a simple solution is sufficient and it is not required to define a complex ontology with reasoning support.

6.4 What's next?

The Interactive Knowledge Stack was a vision that postulated a tight integration of content generation and knowledge-based application building. That vision had to face a number of "reality checks":

(1) "Semantic user interface": we had hoped for graphical user interfaces that could be parameterized at domain entity level, i.e. where there would be an API that connected a domain-specific language with a domain-specific user interface. The only application that got some way towards this objective was the ambient intelligent bathroom and even there, critics may say that too much is still hard coded and where it demonstrates flexibility, too much effort needs to be spent to achieve it.

Our real progress for the user interface came at a level where we did not expect it: in the *VIE* and *create* modules, we used interface features of the evolving HTML5 specification and combined them with a number of existing open source javascript libraries such *jquery* and *backbone.js* that are popular with web developers, but do not rank high on anybody's research agenda. When put together, one ends up with a semantics aware front-end that can communicate well with any RDF-aware backend and, suddenly, we had a relatively tight, yet flexible connection between web-based content management and semantic technologies using RDFa.

(2) "Programming environment for knowledge based content": The principal investigator had hoped for some streamlining in OWL-based reasoning tools, either initiated by IKS or coming from elsewhere in Semantic-Web-Land. This did not happen in the lifetime of the project and the AI-related groups in the semantic web remain entrenched in their partial solutions that still fragment the field and that keep industry as cautious and skeptical

as before. In Stanbol, the semantic technologies group at CNR managed to modularize their own ontology and reasoning “monolith” (KReS) thus giving IKS a more flexible set of tools for actually adding reasoning facilities to content management. We cannot claim that OWL-based reasoning has now entered CMS in a big way, as a result of IKS, but we have opened routes for adding RDF- and OWL-based semantics to CMS thus making CMS vendors fit for linked data and improving the methodology for linking CMS with reasoners.

(3) “Tightly coupled interaction”: The vision of a tightly coupled stack met with opposition from the developers’ quarter. They argued for loose coupling as the only way of delivering web-scale interoperation services. As a result, the standard mode of operation in IKS is RESTful interfaces, but we managed to convince the development team that OSGi bundling would also help. This is the tightest form of coupling that IKS offers.

(4) “Statistics or logical inference”: most CMS vendors have to address content management at web-scale dimensions. In particular, the Web has developed hugely *search-centric*, because the Web’s major characteristic is its *federatedness*. This sounds like a contradiction in terms, but it is not: the large amount of data and the little amount of structure that characterizes the Web, *requires* large-scale search facilities, and search engines from AltaVista to Google have shown that there is a large demand and that large scale search can, not only be done, it is also to date, the winning formula over any schemes that require more structure and more rules to be followed. This has resulted in IKS exploring how NLP tools can be used to “lift” web content to more structure, so as to make it amenable to rule-based and logic-based inference.

To summarize the state of the art, as seen after the IKS project:

(a) the combination of a practical, HTML5 based user interface with some widely adopted libraries that connect web content with RDFa, has helped to move semantic models into traditional content management, at affordable cost to developers.

(b)Semantic web researchers have begun to understand the impedance mismatch between the semantic web layer cake and real software architectures for web-scale content management.

(c) RESTful rules.

(d) Search rules.

Those of us looking for new research challenges should perhaps carefully look at (d): is large-scale search the only paradigm in which WWW-scale information sharing can be supported? We may add another question to this: Will search still work when billions of devices start emitting billions of streams of data, over that same Web-infrastructure?

References

[ABMP08] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. Rdfa in xhtml: Syntax and processing. a collection of attributes and processing rules for extending xhtml to support rdf. w3c recommendation. Technical report, The World Wide Web Consortium (W3C), October 2008.

[ADI08] Ben Adida. hGRDDL: Bridging microformats and RDFa. Journal of Web Semantics, 6(1):54–60, 2008.

[ALLS07] John Allsopp, 2007. Microformats: Empowering Your Markup for Web 2.0. Friends of ED, March 2007.

[ANSW10] What are the success stories of the Semantic Web/ Linked Data? Online available: http://answers.semanticweb.com/questions/1533/what-are-the-success-stories-of-the-semantic-weblinked-data

[AUER06] Auer, S., Dietzold, S., Riechert, T. (2006). OntoWiki – A Tool for Social, Semantic Collaboration. In proceedings of the 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USA, pp. 736-749

[BAT89] Bateman, J. A.; Kasper, R. T.; Moore, J. D.; Whitney, R. A. A General Organization of Knowledge for Natural Language Processing: The Penman Upper Model. USC/Information Sciences Institute. Marina del Rey. 1989.

[BER11] “Decoupling Content Managament” (2011), professional blog entry, http://bergie.iki.fi/blog/decoupling_content_management/)

[BIZE09] Bizer, C. The Emerging Web of Linked Data. IEEE Information Systems. 2009

[BERN01] The Semantic Web. Scientific American, Vol. 284, No. 5, pp. 34–43.

[BLO09] Eva Blomqvist. Ontocase-automatic ontology enrichment based on ontology design patterns. In The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings, volume 5823 of Lecture Notes in Computer Science, pages 65–80. Springer, 2009.

[CIM06] P. Cimiano. Ontology Learning and Population from Text - Algorithms, Evaluation and Applications. Springer, 2006.

[COCK02] A. Cockburn, Agile Software Development, Addison-Wesley, 2002.

[CON07] Dan Connolly. Gleaning resource descriptions from dialects of languages (grddl). World Wide Web Consortium, Recommendation REC-grddl-20070911, September 2007.

[CRUT06] Crutzen, C. K. M. "Invisibility and the Meaning of Ambient Intelligence," International Review of Information Ethics (6), 2006, pp. 1-11.

[COOK09] Cook, D.J., Augusto, J.C. and Jakkula, V.R. "Ambient intelligence: Technologies, applications, and opportunities," Pervasive and Mobile Computing (5), 2009, pp. 277-298.

[CP97] Peter Clark and Bruce Porter. Building concept representations from reusable components. In Proceedings of AAAI'97. AAAI press, 1997.

[CBHS04] Carroll J.J., Bizer C., Hayes P., Stickler P.. Named Graphs, Provenance and Trust. HP Lab; 2004.

[CODD70] Codd EF. A relational model of data for large shared data banks. Commun ACM 1970;13(6):377–387.

[CUN02b] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia, July 2002

[DECK00] The Semantic Web: The Roles of XML and RDF. In IEEE Internet Computing. Vol. 4, No. 5, pp. 63-74.

[DFV02] John Davies, Dieter Fensel, and Frank van Harmelen, editors. On-To-Knowledge: Semantic Web enabled Knowledge Management. J. Wiley and Sons, 2002.

[DKDA05] L. Ding, P. Kolari, Z. Ding, S. Avancha, T. Finin, A. Joshi, 2005. Using Ontologies in the Semantic Web: A Survey. TR CS-05-07.

[DKFJ05] Ding L., Kolari P., Finin T., Joshi A., Peng Y., Yesha Y.. On Homeland Security and the Semantic Web: A Provenance and Trust Aware Inference Framework. In: Proceedings of the AAAI Spring Symposium on AI Technologies for Homeland Security; 2005.

[DOBA12] M. Dow, S. Bayliss, 2012. Fedora Commons embedded semantic services using Apache Stanbol. In Proceeding of the 7th International Conference on

Open Repositories, OR2012. Online available: https://www.conftool.net/or2012/index.php?page=browseSessions&form_session=68

[ERI07] Henrik Eriksson. The semantic-document approach to combining documents and ontologies. Int. J. Hum.-Comput. Stud., 65(7):624–639, 2007.

[EUR13] “Commission unveils new translation engine as job cuts loom”, February, 2013, http://www.euractiv.com/culture/commission-unveils-new-translati-news-518050

[FGPJ97] M. Fernández, A. Gómez-Pérez, and N. Juristo. Methontology: from ontological art towards ontological engineering. In Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, 1997.

[FLGP02] M. Fernández López and A. Gómez-Perez, 2002. Overview and analysis of methodologies for building ontologies. The Knowledge Engineering Review, 17, pp. 129-156, doi:10.1017/S0269888902000462

[GMB08] Aurona Gerber, Alta van der Merwe, and Andries Barnard. 2008. A functional semantic web architecture. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications* (ESWC'08), Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis (Eds.). Springer-Verlag, Berlin, Heidelberg, 273-287.

[GP09] Aldo Gangemi and Valentina Presutti. Ontology design patterns. In Handbook on Ontologies, 2nd Ed., International Handbooks on Information Systems. Springer, 2009.

[GR08] Good Relations: A Web Vocabulary for E-Commerce. “GoodRelations in the Wild”. Online available: http://wiki.goodrelations-vocabulary.org/Datasets

[GRU93] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, Formal Ontology in Conceptual Analysis and Knowledge Representation, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[GRU09] Thomas R. Gruber. Ontology. In Encyclopedia of Database Systems, pages 1963–1965. Springer-Verlag, 2009.

[GRU93] T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993. Available online: http://tomgruber.org/writing/ontolingua-kaj-1993.htm

[GRU09] T.R.Gruber. Ontology. In the Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2009. Online available: http://tomgruber.org/writing/ontology-definition-2007.htm

[GRÜN94] Grüninger, M.; Fox, M. S. The Role of Competency Questions in Enterprise Engineering. IFIP WG 5.7 W orkshop on Benchmarking. Theory and Practice. Trondheim, Norway. 1994.

[HAYE04] P. Hayes. RDF Semantics. http://www.w3.org/TR/2004/ REC-rdf-mt-20040210/; 2004.

[HCHD11] Heitmann et al., "An empirically grounded conceptual architecture for applications on the Web of Data", IEEE Transations on Systems, Man and Cybernetics – Part C: Applications and Reviews, 2011.

HEBI09] Heath, T., Bizer, C. Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on The Semantic Web Theory and Technology. Online available: http://linkeddatabook.com/editions/1.0/

[HEHU03] Hefflin, J. and Huhns, M. N., 2003. The Zen of the Web. In IEEE Internet Computing. Vol. 7, No. 5, pp. 30-33.

[HEND01] Agents and the Semantic Web. In IEEE Intelligent Systems. Vol. 16, No. 2, pp. 30-37.

[HPPH05] Semantic Web Architecture: Stack or Two Towers? Horrocks, Ian; Parsia, Bijan; Patel-Schneider, Peter F; Hendler, James A, In: Fages, François; Soliman, Sylvain. PPSWR: Principles and Practice of Semantic Web Reasoning, Third International Workshop, PPSWR 2005, Dagstuhl Castle, Germany, September 11-16, 2005, Proceedings; Springer; 2005. p. 37-41.

[HOSA02] I. Horrocks, U. Sattler. Description Logics Basics, Applications, and More. Tutorial at ECAI-2002, http://www.cs.man.ac.uk/ ~horrocks/Slides/ecai-handout.pdf; 2002

[HUYN07a] Huynh, D. F., Karger, D. R., & Miller, R. C. (2007). Exhibit: lightweight structured data publishing. In Proceeding of the 16th International Conference on World Wide Web, Banff, Alberta, Canada, pp. 737-746.

[HUYN07b] Huynh, D. F., Karger, D. R., Miller, R. C. Potluck. (2007). Data Mash-Up Tool for Casual Users. In Proceeding of the 6th International Semantic Web Conference, Busan, Korea, pp. 239-252.

[IKS-D2.2] F. Christ, G. Engels, S. Sauer, G.B. Laleci, E. Alpay, T. Namli, A.A. Sinaci, F. Tuncer, 2009. IKS Deliverable - Report 2.2. “Requirements Specification for the Horizontal Industrial Use Case”. Online available: http://www.iks-project.eu/sites/default/files/iks-d22-requirements-horizontal-case-20100304.pdf

[IKS-D3.1] M. Romaneli et al., 2010. IKS Del. D3.1 “Model of Knowledge Based Interaction”. Online: http://www.iks-project.eu/resources/model-knowledge-based-interaction

[IKS-D3.2] A. Adamou et al., 2010. “Ontological Requirements for Industrial CMS Applications”. IKS Project IKS Del. D3.2. Online: http://stlab.istc.cnr.it/documents/iks/IKS-D3.2.pdf

[IKS-D4.1] S. Janzen, E. Blomqvist, A. Filler, S. Gönül, T. Kowatsch, A. Adamou, S. Germesin, M. Romanelli, V. Presutti, C. Cimen, W. Maass, S. Postaci, E. Alpay, T. Namli, G. Banu Laleci Erturkmen. IKS Deliverable – D4.1 Report: AmI Case Design and Implementation (Public), 2011.

[IKS-D4.2] F. Christ, A.A. Sinaci, S. Gonul, G. Engels, B. Nagel, S. Sauer, O. Grisel, R. Kurz, 2012. D.4.2. Horizontal Industrial Use Case Design and Implementation. Online available: http://www.iks-project.eu/sites/default/files/iks_d42_horizontal_industrial_use_case_20120229.pdf

[IKS-5.0-Alpha] F. Christ, G. Engels, B. Nagel, S. Sauer, S. Germesin, E. Daga, O. Kilic, 2010, IKS Alpha Development. Online available: http://wiki.iks-project.eu/index.php/IKS_Alpha_Development

[IKS-D5.0] IKS Deliverable 5.0. Design anf Implementation of Interactive Knowledge Stack (Compendium). F. Christ at all., March, 2012. Online: http://www.iks-project.eu/sites/default/files/iks_del_5_0_design_implementation_IKS_compendium_2012.pdf

[IKS-D5.1] IKS Deliverable 5.1. Intermediate Report – Interaction and Presentation.

[IKS-D5.2] IKS Deliverable 5.2. Intermediate Report – Knowledge Representation and Reasoning.

[IKS-D5.3] IKS Deliverable 5.3. Intermediate Report – Semantic Lifting.

[IKS-D5.4] IKS Deliverable 5.4. Intermediate Report – Semantic Data Access and Persistence Components.

[IRS09] L. Iannone, A. Rector, R. Stevens. Embedding knowledge patterns into owl. In 6th Annual European Semantic Web Conference (ESWC2009), pages 218–232, June 2009.

[JEBJ03] J. Golbeck, B. Parsia, J. Hendler. Trust Networks on the Semantic Web. In: Proceedings of Cooperative Intelligent Agents; 2003.

[KAC96] The KACTUS Booklet version 1.0. Esprit Project 8145. September, 1996. (http://www.swi.psy.uva.nl/prjects/ *NewKACTUS/Reports.html*

[KNI94] K. Knight, S. Luck. Building an Large Knowledge Base for Machine Translation. Proceedings of the American Association of Artificial Intelligence Conference (AAAI- 94). Seattle (USA). 1994.

[KNI95] K. Knight, I. Chancer, M. Haines, V. Hatzivassiloglou, E.H. Hovy, M. Iida, S.K. Luk, R.A. Whitney, K. Yamada, Filling Knowledge Gaps in a Broad- Coverage MT System. Proceedings of the 14th IJCAI Conference. Montreal (Canada). 1995.

[KW-D1.4.2v2] J.Z. Pan, Lancieri, L., Maynard, D., Gandon, F., Cuel, R., Leger, A. Deliverable 1.4.2v2. Success Stories and Best Practices. Knowledge Web project. http://knowledgeweb.semanticweb.org/semanticportal/deliverables/D1.4.2v2.pdf

[LAMG01] O. Lassila, D. McGuinness. The Role of Frame-Based Representation on the Semantic Web. Stanford University; 2001.

[LCP05] V. Lanfranchi, F. Ciravegna, D. Petrelli. Semantic web-based document: Editing and browsing in aktivedoc. In Proceedings of the 2nd European Semantic Web Conference , Heraklion, Greece, May 29-June 1 2005.

[LEE06] T.B.Lee, 2006. Linked Data. Online available: http://www.w3.org/DesignIssues/LinkedData.html

[LVHN05] T. Liebig, F. W. von Henke, O. Noppens. Explanation support for owl authoring. In Thomas Roth-Berghofer and Stefan Schulz, editors, ExaCt, volume FS-05-04 of AAAI Technical Report, pages 86–93. AAAI Press, 2005.

[MAMI06] Maeda, E. and Minami, Y. "Steps towards Ambient Intelligence," NTT Technical Review (4), 2006, pp. 50-55.

[MAGR11] G. Madhu, A. Govardhan, T.V. Rajinikanth, 2011. Intelligent Semantic Web Search Engines: A Brief Survey. International journal of Web & Semantic

Technology (IJWesT) Vol.2, No.1, January 2011. Online: http://arxiv.org/abs/1102.0831

[MARP03] R. Matthew, A. Rakesh, D. Pedro. Trust Management for the Semantic Web. In: Proceedings of the Second International Semantic Web Conference; 2003.

[MIK09] P. Mika. Year of the Monkey: Lessons from the First Year of Searchmonkey. In Stefan Fischer, Erik Maehle, and Rüdiger Reischuk, editors, GI Jahrestagung, volume 154 of LNI, page 387. GI, 2009.

[MVH04] D.L. McGuinness & F. van Harmelen. OWL web ontol- ogy language overview. W3C recommendation, W3C, 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[NRB09] N. Nikitina, S. Rudolph, S. Blohm. Refining ontologies by pattern-based completion. In E. Blomqvist, K. Sandkuhl, F. Scharffe, V. Svatek (Eds.), Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with the 8th ISWC 2009, Washington D.C., Vol. 516. CEUR Workshop Proceedings, 2009.

[OP4L-D1.1] Social Semantic Web technologies and tools and their educational applications. Online available: http://op4l.fon.bg.ac.rs/sites/default/files/OP4LD1.1.pdf

[OREI05] O'Reilly, T. (2005). What Is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software. Online available

[PATS05] Peter F. Patel-Schneider: A Revised Architecture for Semantic Web Reasoning. PPSWR 2005: 32-36.

[PDGB09] V. Presutti, E. Daga, A. Gangemi, E. Blomqvist. eXtreme Design with content ontology design patterns. In E. Blomqvist, K. Sandkuhl, F. Scharffe, V. Svatek (Eds.), Proceedings of the Workshop on Ontology Patterns (WOP 2009), Washington D.C., Vol. 516. CEUR Workshop Proceedings, 2009.

[POP02a] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff and M. Goranov, 2004. KIM -- Semantic Annotation Platform, Journal of Natural Language Engineering.

[PTSS04] H. Sofia Pinto, Christoph Tempich, Steffen Staab, and York Sure. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engingeering of ontologies. In Ramon L´opez de M´antaras and Lorenza Saitta, editors, Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), August 22nd - 27th, pages 393–397, Valencia, Spain, AUG 2004. IOS Press.

[RAY12] Rayfield, J. 2012. “Sports Refresh: Dynamic Semantic Publishing”. Online available at: http://www.bbc.co.uk/blogs/bbcinternet/2012/04/sports_dynamic_semantic.html

[RWW10] R. Mac Manus, 2010. How Best Buy is using the Semantic Web. Source: ReadWrite Web. Online available: http://readwrite.com/2009/12/02/top_10_semantic_web_products_of_2009

[SCH95] G. Schreiber, B. Wielinga, W. Jansweijer. The KACTUS View on the ‘O’ World. In Proceedings of the National Dutch AI Conference. NAIC’95. 1995.

[SCHH04] P. Patel-Schneider, P. Hayes, I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. http://www.w3.org/TR/2004/ REC-owl-semantics-20040210/; 2004.

[SGUC03] P. da Silva Pinheiro, D. McGuinness, R. McCool. Knowledge Provenance Infrastructure. Data Engineering Bulletin 2003; 26(4): 26–32.

[SOWA02] J. Sowa. Semantic Networks. http://www.jfsowa.com/pubs/ semnet.htm (last modified: 08/12/2002 12:18:06); 2002.

[SFGP09] M.C. Suárez-Figueroa & A. Gómez-Pérez. NeOn methodology for building ontology networks: a scenario-based methodology. In Services & Seman- tic Technologies (S3T 2009) International Conference on Software, editor, Proceedings of the International Conference on Software, Services & Semantic Technologies (S3T 2009), 2009.

[SOWA06] J. Sowa. The Challenge of Knowledge Soup. In *Research Trends in Science, Technology and Mathematics Education,* edited by J. Ramadas & S. Chunawala, Homi Bhabha Centre, Mumbai, 2006.

[SS02] Y. Sure & R. Studer. On-To-Knowledge methodology. In Davies et al. [DFv02], chapter 3, pages 33–46.

[STJO09] Stankovic, M., and Jovanovic, J. (2009). TagFusion - A System for Integration and Leveraging of Collaborative Tags," Annals of Information

Systems, Special Issue on Semantic Web & Web 2.0. Springer-Verlag, Berlin, Germany, pp. 3-23.

[SWA97] Swartout, B.; Ramesh P.; Knight, K.; Russ, T. Toward Distributed Use of Large-Scale Ontologies. Symposium on Ontological Engineering of AAAI. Stanford (California). Mars, 1997.

[SW.COM10] “What are the success stories of the Semantic Web/ Linked Data?” Online available at semanticweb.com: http://answers.semanticweb.com/questions/1533/what-are-the-success-stories-of-the-semantic-weblinked-data (last access: October 23, 2012)

[USC95] Uschold, M., King, M. Towards a Methodology for Building Ontologies. Workshop on Basic Ontological Issues in Knowledge Sharing. 1995.

[VHTTW09] Frank van Harmelen, Annette ten Teije, and Holger Wache. Knowledge engineering rediscovered: towards reasoning patterns for the semantic web. In Proceedings of the 5th International Conference on Knowledge Capture (K-CAP 2009), September 1-4, 2009, Redondo Beach, California, USA, pages 81–88. ACM, 2009.

[VPST05] Denny Vrandecic, H. Sofia Pinto, York Sure, and Christoph Tempich. The diligent knowledge processes. Journal of Knowledge Management, 9(5):85–96, October 2005.

[YOVA02] G. Yolanda, R. Varun. Trusting Information Sources One Citizen at a Time. In: Proceedings of International Semantic Web Conference 2002, pp. 162–176. 2002

[ZHYU04] D. Zhongli, P. Yun. A Probabilistic Extension to Ontology Language OWL. In: Proceedings of the 37th Hawaii International Conference On System Sciences (HICSS-37). Big Island, Hawaii; 2004.

[ZHYR04] D. Zhongli, P. Yun, P. Rong. A Bayesian Approach to Uncertainty Modelling in OWL Ontology. In: Proceedings of 2004 International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA2004). Luxembourg-Kirchberg, Luxembourg; 2004.

iks-project.eu

The IKS Handbook

iks.project.eu

ISBN: 978-3-902448-35-4
2013