# Approaches and Tools for Viewing, Browsing and Querying Semantic Web Data and Ontologies

Tatyana Ivanova

**Abstract**—Semantic Web is not only a vision, it is the future of the Web, and current Web already includes many well working semantic-based applications. Semantic Web is mainly for computers to make web more convenient for humans, but many peoples have need in direct usage of semantic web knowledge. Thus, querying semantic web and visual representation of semantic web knowledge is of great importance. In this paper we make a brief survey and classification of current semantic web technologies and querying tools. Our main aim is to propose architecture that can support recommendation and easy integration of the right tools for every type user or purpose among the grand amount of the available well working tools for knowledge exploration evolution and querying semantic web.

**Index Terms**—ontology, ontology modelling languages; Semantic Web querying languages; ontology visualization tools; ontology querying tools

## 1 Introduction

SEMANTIC Web (SW) vision is to make the Web more convenient for users by facilitating machines to perform more qualified search and information integration. Semantic Web relies heavily on the linked data and formal ontologies that structure web content for the purpose of comprehensive and transportable machine processing. Ontologies are used to describe the semantics of web resources and support machines by proposing needed information in machine readable and processable formats. Many Semantic web languages have been developing recently to facilitate modeling of linked data and semantic representation of knowledge in various domains. A thousands of ontologies are stored in ontology libraries, a grand amount of linked data are accessible through SPARQL endpoints. Several Semantic Web Querying languages have been developing recently to facilitate extraction of linked data, metadata and knowledge, stored in the Web. Unfortunately, the use of this knowledge is too limited for a number of reasons.

In our work, after the brief analysis of Semantic web Languages, linked data and ontology querying languages and tools, we propose a flexible and modular architecture, that can support ontology visualization, testing, evolution, querying of linked data and extraction of knowledge from ontologies for all the type SW users (knowledge professionals, domain experts, nonprofessionals, software agents).

The paper is structured as follows:

The first section make a brief survey of the most usually used Semantic Web and Ontology modelling languages, as querying technologies are in thight dependence of the used data, metadata and knowledge modeling technologies.

In the second section we make a brief survey of the most usually used Semantic Web querying languages. The goal of this survey is to answer the question how to make querying semantic web data and knowledge more easy for different type users. As direct usage of querying languages requires deep understanding of semantic web technologies and skills in writing queries in many languages, following strict synthactical rules, graphical querying tools are of great importance.

In the third section we survey visualization and assistance tools, used in the process of understanding and querying Semantic Web by different type users, and conclude that Semantic Web developers need to have grand variety of different visualization and querying tools for effective development and testing of semantic web knowledge and tools. To satisfy these needs, we propose the integrated ontology development and querying architecture.

## 2 Ontology Modelling Languages

There are a grand variety of formalisms, used for data and knowledge representation in Semantic Web. According to the underlined models we can classify Semantic Web languages in three main classes: frame–based, RDF triples–based and Logic-based languages (Figure 1).

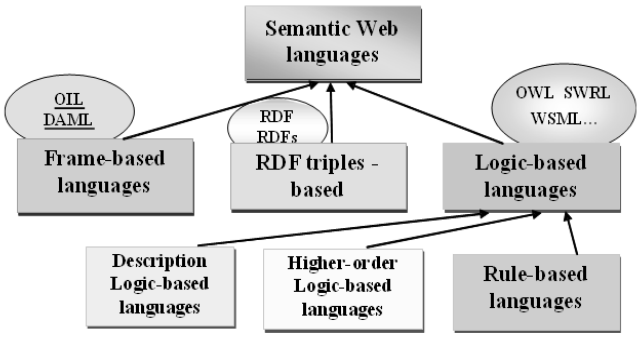



Fig. 1. Classification of Semantic Web languages

Frame-based formalisms, as OIL or DAML are the oldest, and less frequently used. Frame-based modelling languages use frames for knowledge representation. They were used for knowledge modeling in artificial intelligence in 20 century. Frame-based models are similar to class hierarchies in object-oriented programming languages, but there are several differences between the two models: while frames are focused on explicit and intuitive representation of knowledge, objects focus is on encapsulation and data hiding. In object-oriented programming every object is an object of only one class, whereas in frame–based formalisms one and the same instance

may be the instance of several classes.

RDF (Resource Description Framework) was one of the oldest Semantic Web languages, developed to represent linked data in a machine-processable way. It is widely used in the Web. RDF is based on a tree-like graphical formalism, and usually is serialized by XML-based syntax. The RDF model includes hierarchically structured triples (subject, predicate, Object). RDFS (RDF Schema) extends RDF with a schema vocabulary that allows users to define basic terms such as types, classes, properties, ranges, domains, and the relations between them. RDFS is a simple vocabulary language for expressing the hierarchical and other semantic relationships between resources.

OWL (Web Ontology Language) allows users to represent richer semantics of knowledge in a machine-processable way. It is the proposed Semantic Web standard for building Web ontologies. OWL uses RDF and RDF Schema, and extends them by defining types of relationships, constructions for compound classes, and several types of properties for specifying relationships between different resources. It has strong formal semantics with roots in DL. OWL is RDFS extension, having stronger syntax and a much richer, more expressive vocabulary for defining Semantic Web ontologies. OWL has three sublanguages: OWL Lite, OWL DL and OWL full. OWL Lite is the simplest OWL sublanguage with minimal reasoning capabilities and maximal effectiveness. It usage is easier, and is recommended for logically simple domains. OWL DL includes OWL Lite and has DL reasoning capabilities. It realizes decidable restriction of FOL and can guarantee maximal reasoning power and the decidability at the same time.

OWL full includes OWL DL and is union of OWL syntax and RDFs semantics with no expressiveness constraints. Unfortunately, it can not guarantee decidability and reasoning effectiveness is unpredictable.

OWL2 is an revision and extension of the OWL Web Ontology Language, which adds useful elements to the current OWL such as property chains, keys, data ranges, and richer data types. It also has three Profiles: OWL2 RL, OWL2 QL, and OWL2 EL. OWL2 RL is useful for applications that require scalable reasoning without too much expressive power, and where query answering is most important. OWL2 EL is useful for applications, using richer hierarchies (grand number of properties and/or classes); OWL2 QL is useful for applications with very large amount of instances. There is also OWL 2 full, having RDFs – based semantics and maximal reasoning power, which (as OWL full) is undesidable.

There are also many other Semantic Web languages as SWRL, RIFF, WSML, etc. All these languages are needed because of the grand variety of knowledge structures and significant difference in reasoning methods, used in different real world domains. No one of these languages can replace the other and be accepted as the only universal Semantic Web language Two and more of these languages are frequently used in one and the same project. The different query languages are needed for querying linked data or ontologies, represented by different ontology languages. We will survey Semantic Web querying languages

## 3 Semantic Web Querying Languages

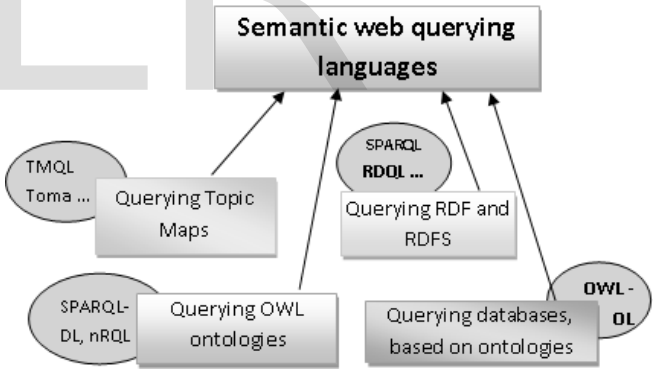
Semantic Web querying languages play a central role in extracting and processing information about the real world from Linked data and ontologies. As each ontology language provides different expressive power and computational complexity for reasoning, and requirements of the semantic web application, related to knowledge extraction are very different, various ontology query languages have been developed to query the information defined by different ontology languages, and to provide users with the ability to retrieve various type information from the ontologies. Several of these query languages are implemented in ontology query tools and systems.

Semantic Web querying languages can be classified according to different criteria. We will make a brief analysis of querying languages according to important criteria such as underlined semantics, syntax, and purpose. According to the underlined semantics query languages can be divided into four main types:

1. Path – based;
2. Based on RDF triples;
3. Based on graphs and;
4. Using logic elements.

According to the type of ontologies, to which query languages are designed they can be classified as follows (Fig.2):

1. Languages querying Topic Maps;
2. Languages querying RDF and RDFS documents;
3. Languages querying OWL ontologies;




4. Languages querying databases, based on ontologies.

Fig 2. Classification of Semantic Web querying languages

Topic Map Query Languages are an XML-based extension of Structured Query Language (SQL). Several such languages are proposed with four main type underlined models: path-based languages, SQL-inspired languages (as TMRQL), functional languages (as AsTMA?), and languages including logical elements (as Tolog). The main elements of the model of the path-based languages (such as TMQL, Toma, TMPath) are nodes and arcs. Typically queries are executed by going around nodes, using roads, presented by arcs.

One of the most widely used formalism for querying RDF and RDFS documents is SPARQL (SPARQL Protocol and RDF Query Language). It is both SPARQL Protocol and family of RDF Query Languages. Syntactically SPARQL family languages are SQL-like languages for querying RDF graphs, based on matching graph patterns. The simplest graph pattern

is the triple pattern, which corresponds to a RDF triple (subject – predicate - object triple). RDF queries can contain a variable instead of an RDF term in one of these three positions.

SPARQL family languages are very good for extracting RDF data and for queries in a networked, web environment. The SPARQL drawbacks are its immaturity, the lack of support for transitive/hierarchical queries, the lack of reasoning capabilities, and lack of wide standardized deployment. SPARQL has two main versions: SPARQL 1.0 and SPARQL 1.1.

RDF-based query languages (RDQL, SeRQL, SPARQL) can also be used to query ontologies, but they are quite hard to give a semantics w.r.t OWL-DL and at the same time are more expressive than OWL-DL reasoners can cope with. On the other hand, query languages based on Description Logics (RQL, DIG, nRQL) have clearer semantics but are not powerful enough. More sophisticated query languages, used in ontology querying, combine direct data and schema extraction and reasoning over queries, or extracted components.

SPARQL-DL was introduced in [2] as a subset of SPARQL with clear OWL-DL based semantics and more expressive than existing DL query languages and able to be implemented on top of existing OWL-DL reasoners like Pellet.

OWL Query Language OWL-QL [3] is a well designed language for querying over knowledge represented in a ontology repository. OWL-QL is an updated version of DAML Query Language (DQL). Results from OWL-QL query are produced by applying the bindings to the query pattern and considering the remaining variables in the query pattern to be existentially quantified. Subontologies also may be extracted as answers. OWL-QL supports query-answering dialogues in which the answering agent may use automated reasoning methods to derive answers to queries, as well as dialogues in which the knowledge can be used for answering a query.

There are also some other Semantic Web query languages, profiles, versions. All these languages are needed because of the grand variety of Semantic Web languages and low efficiency of built in query languages reasoning capabilities. The different query languages are needed for querying linked data or ontologies, represented by different ontology languages, or for different needs, related to reasoning over extracted data and metadata. Many SPARQL endpoints don't support complicated features to ensure higher effectiveness. To ensure good balance between reasoning needs and efficiency, queering tools should be very flexible. They should support several query languages, sublanguages, profiles, and propose easy to use interface, for different type of users, that hide in the greatest possible extent the syntactic diversity.

## 4 Tools for Extracting Data or Metadata from Semantic Web Documents

It is clear, that different tools are needed for different type users, different knowledge representation formats and different searching tacks. For example, software agents don't need any graphical interface, whereas knowledge engineers need good ontology visualization components. And when user can extract only stored in the store data, reasoning capabilities will be useless and would only reduce the efficiency.

We will first briefly survey different tools, needed for querying semantic web documents, and then will discuss how to select the right tool. In the concept of querying tools we include not only tools for building query, returning and showing results, but also tools, storing data or onologues that provide stored results.

We will classify query management tools according to important criteria such as target type users, purpose, and usual location.

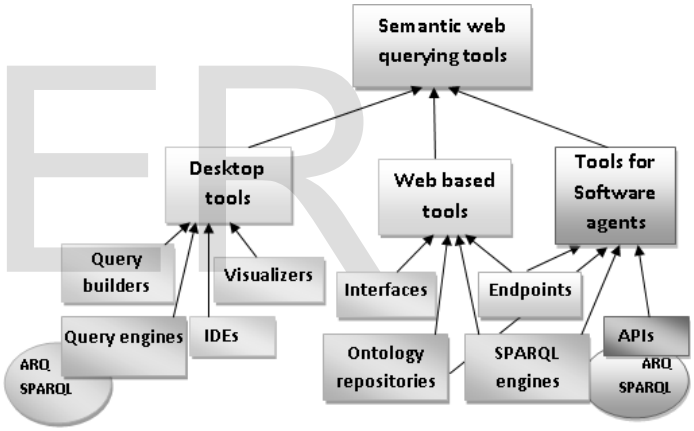
According to the target type users query tools can be divided into two main types:

1. For software agents;
2. For humans.

And different type of tools are needed for humans, having different level of understanding of Semantic web technologies (as knowledge engineers, domain experts, non professional users), or for different purposes (testing ontologies, extracting data, metadata, including reasoning).

According to the usual location of querying tools, they can be divided into two main types: Desktop tools and Web tools (figure 3).

Fig. 3. Classification of Semantic Web querying tools




Desktop tools are components of integrated semantic web development environments, or highly specialized. Main type tools are:

1. Visualizers;
2. Querying tools;
3. Components of ontology IDEs for visualization;
4. Components of ontology IDEs for querying;
5. Query processors;
6. Basic query Processors (SPARQL ARQ for Jena);
7. Distributed processors;
8. Tools for domain – specific knowledge visualization or management.

Web - based tools may have in some cases similar purpose, but there are some differences in used underlined technologies and working mechanisms. Distributed web technologies for open and insecure environments are used both for interface building and core functionality. Main type tools are:

1. User interfaces;
2. Endpoints;
3. SPARQL Processors;

4. Ontology libraries.

Additional tools, as reasoners, language translators, and file converters can be used as components of every type matured tools. We will first discuss tools, used by software agents in semantic web querying.

Agents usually don't need any visualization of ontologies or extracted results. If it's users need some visualization of ontology or results, the software agent should do it or start and use some visualization tool.

### 4.1 Semantic Web Querying and Software Agents

Software agents need specific APIs to query ontologies. Usually such APIs include class's methods for creation and manipulation of ontology models. Using appropriate APIs and web services, agents have three main approaches to query ontologies:

1. By sending SPARQL – like queries and processing returned results;
2. By loading ontologies from ontology libraries and using his own API to extract needed information;
3. By searching ontologies, loading them and using his own API to extract needed information.

Ontology query APIs usually are components of some library for ontology creation and management programmatically, called Ontology API. There are several such libraries: OWL2 API, Jena Ontology API, SOQA, Manchester ontology API, etc. Some ontology query APIs can work only with ontologies, serialized in specific languages, and others are language - neutral.

ARC is a simple and flexible API, enabling combination of the microformats with RDF solutions. ARC includes Parsers for RDF/XML, Turtle, SPARQL + SPOG, RSS. It can work with N-Triples, RDF/JSON, RDF/XML, Turtle formats.

Sesame is an open-source framework for analyzing and querying RDF data. The Sesame Rio (RDF Input/Output) package contains a Java based API for RDF parsers, writers and Alibaba, an API for mapping Java classes onto ontologies. Sesame supports two query languages: SPARQL and SeRQL.

OWL2 API include RDF/XML parser and writer, OWL/XML parser and writer, OWL Functional Syntax parser and writer, Turtle parser and writer, KRSS parser, OBO Flat file format parser, Reasoner interfaces for working with reasoners such as FaCT++, HermiT, Pellet and Racer.

The Jena Ontology API is directed to RDF and OWL – based languages and related query languages. It provides a consistent programming interface for ontology application development, independent of which ontology language is used in programs. Jena includes support for a variety of reasoners through the inference Jena Ontology API, for example can work with RDFS and all the OWL 1 profiles, but don't support OWL 2 profiles.

There is also ontology querying APIs that can query ontologies in several languages. [1] presents an ontology language independent Java API SOQA (SIRUP Ontology Query API). This independence is achieved by using metamodel, representing modeling capabilities that are typically supported by several ontology languages to describe ontologies and their components (concepts, attributes, methods, relationships, instances, ontology metadata).

Software agents may also use reasoners to enhance SPARQL–like languages capabilities. Reasoners can be used in the data store to infer additional triples and store them in memory before running query, or before the endpoint to process a data set, and add any inferred triples to the returned results. Pellet, for example supports the SPARQL-DL format for querying OWL DL and OWL 2 ontologies. Pellet can be accessed via three different APIs: the Manchester OWL API, the internal Pellet API, and the Jena API. The internal API is the most effective, but is incomplete and has low usability. The Manchester OWL API is an OWL-oriented API with a wide range of features for managing ontologies but does not support SPARQL. Jena is a very popular, rich and stable API but lacks specific OWL 2 support.

Distributed query processors analyze all triple patterns in a query, rewrite an initial query in a federated form by using analyzer results, send result subqueries to appropriate endpoints of datasets and merge all query results received from datasets. WoDQA is an example of federated query based tool and can discover all possible related datasets for a query and transform initial queries to federated queries which will be executed over distributed datasets. SPARQLBot is another web-based service that reads and writes Semantic Web data. SPARQLBot can process microformats, RSS, several RDF serializations, and results from parameterized SPARQL queries.

### 4.2 Querying Semantic Web by Humans

Semantic Web is mainly for software, but in many cases humans need to query semantic web data: for example to find the right source of knowledge or to test the quality of developed ontology. Different type of users may need to query semantic web:

1. lay users;
2. Domain – familiar users;
3. Ontology querying professionals.

To do that, every user needs some interface to view the querying ontology structure, to be assisted in writing complex queries, to view returned results. Two main classes of visualization interfaces are used: graphical and form – based. And graphical representations may be domain – independent (viewing for example dependencies between classes, properties, instances) and domain – specific, included some domain contexts (for example geographical maps).

There are several types of linked data and ontology extracting or viewing tools:

1. Browsers;
2. Visualization tools;
3. Editors;
4. IDEs and IDE visualization components;
5. Domain-specific visualizing tools;
6. Querying tools;
7. Linked data, ontology and query results visualization tools.

### 4.3 Linked Data Browsers

Linked Data Browsers are very useful for viewing the linked data structure before querying it. Usually the first step in data understanding by the user is general overview that obtains a full picture of the data, then searching, filtering and zooming techniques are used to focus on interesting items. Interactive interfaces in every one of these steps (graphical, or form-based), as well as interfaces for viewing of details and statistics

need to be included in linked data browsers. For information retrieval from linked data stores Linked Data Browsers usually use SPARQL queries that are automatically generated as a result of user interaction via form – based or menu-based interface.

There are server side Linked Data Browsers and some web browser extensions, working on the client side. Some browsers are oriented to specific domains, but the most are domain – independent. Example of server-side Linked Data browser is Disco. Open Link Data Explorer is a Web browser extension, and a server-side component of the Open Link Ajax Toolkit. Quick & Dirty RDF Browser and Graphity Browser are domain – independent RDF and RDFa browsers. DBpedia Mobile Linked Data browser is suited for lay users and use DBpedia locations as starting points for exploring the geospatial Semantic Web. Fenfire Linked Data browser displays information as a navigatable graph. Another browser, that use graph representations is IsaViz. Sextant [8] is a tool that enables the visualization and exploration of the spatial dimension of linked geospatial data. Sextant enables map creation and sharing, visualization and exploration of data by evaluating GeoSPARQL queries on SPARQL endpoints.

Rhizomer (Bereta, Nikolaou, Karpathiotakis, Kyzirakos, & Koubarakis, 2013) propose capabilities in data overview, analysis, zoom and filter by easy to use interaction patterns users are already familiar with but that are automatically generated from data and ontologies. Users can interactively explore the data using facets, and combine filters for faceted views to build complex queries. This new approach makes the usual form and menu-based interface very flexible, dynamic, easy to use and suitable for lay users.

### 4.4 Ontology Visualization Approaches and Tools

The context – based visualization is important for querying, as it helps the user to be aware of the knowledge structure and content. Visualization is also needed for making possible usage of the easy for the user drag and drop mechanism in query preparation. As results of some type queries are subontologies of the querying ontology, it is also important to have a graphical representation of the results.

A grand variety of Semantic Web tools have possibilities to visualize semantic web content in various contexts. Different tools can visualize different Semantic Web formats, or use different visualizing approaches. We will first discuss ontology visualization approaches, and then

#### 4.4.1 Ontology Visualization Approaches and Techniques

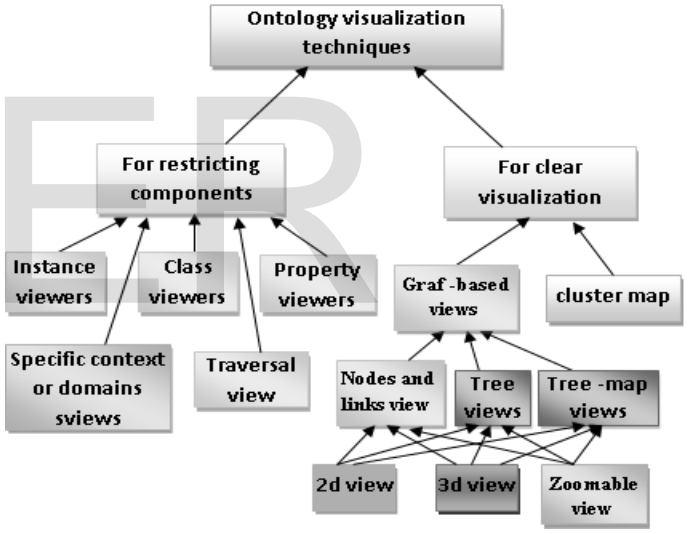
Using the view-based mechanism, the ontology is loaded and manipulated in the view, using all the information required to specify the latest state of the view from combination of the view configuration, the DL ontology, and the instance store.

Users with different knowledge about ontologies have different requirements for ontology visualization. Different views also needed for different tasks, used ontologies. For example, ontology developer's view should include all the technical details of concepts and relations, as in contrast for information searchers the view should be less technical and provide as much information in natural language. Thus various visual knowledge representations should be supported in the ontology browsers, editors or integrated development environments by providing alternative views to ontology components depending on the users needs. Users should switch by the user interface between different properties or visualization approaches.

The view is a set of related ontology components that map to more complex constructs in the underlying ontology. It is a subset specification on an ontology, which allows extracting a manageable portion of the ontology. Usually the view is a specification of a sub-graph of the ontology. Generating "views" over ontologies allows ontologies to be customized for use within specific application contexts. For editable views, when entities are manipulated in the view, corresponding modifications are made automatically in the original ontology. In the other cases, the view is made only for visualization purposes.

Many different views for one and the same ontology may be used. They differ from each other in used approaches for selection of components for visualization, and in used visualization techniques (figure 4). Some of the most popular view types, based on component restriction approaches are schema view, property view, hierarchy view, instance view, Traversal view.

Fig.4. Classification of Ontology visualization approaches and techniques




Traversal View [5] is a view where a user specifies the central concept or concepts of interest, the relationships to traverse to find other concepts to include in the view, and the depth of the traversal. For example, given a large ontology of computer science, a user may use a Traversal View to extract a concept of programming languages and approaches that are related to them. [11] defines the notion of Traversal Views formally, and shows that if the ontology is consistent, every Traversal View also represents a consistent ontology. He also presents a strategy for maintaining the view through ontology evolution and describes a tool for defining and extracting Traversal Views.

Another important dimension of visualization includes used visualization techniques. [14] categorize the most common ontology visualization and presentation techniques in six types: Indented list, Node–link and tree, Zoomable, Space-filling, Focus + context or distortion, and 3D Information landscapes. The cluster map is a visualization technique suitable for simple schemas with Instances viewing.

Cluster maps contain mainly information about the instantiation of the classes, and show the overlaps between them. Instances with the same class membership are grouped in clusters. When subclass relations hold between two classes, the clusters are connected by a directed edge. Visualization of the instances and their relationships shows important information for querying, but in ontologies with a high number of overlaps too many crossing edges lead to a cluttered view.

### 4.4.2 Ontology Editors, Visualization Tools, and Querying Tools

Ontology editors are useful in ontology visualization, and some of them have querying capabilities. There is a long list of ontology editors. We may classify them according to various dimensions: functional, architectural, usability. Main differences between editors are related to the degree to which the editor abstracts from the actual ontology representation language used for persistence, the visual navigation possibilities within the knowledge model, integration of the editors and other tools (for example reasoners and querying tools), web orientation, publishing mechanisms, underlined platform and architecture.

General purpose editors are usable in many domains. Some of them can edit ontologies, serialized in only one language (for example EMFText OWL2 Manchester Editor can use only Manchester syntaxes), but most editors support main semantic Web languages. RDF Gravity (RDF Graph Visualization Tool) for example is a desktop tool for visualizing directed graphs stored in RDF and OWL formats. It provides a simple, flexible, and powerful visualization of RDF graph structures and using filters to visualize specific fragments of RDF Graphs, it also supports Full text Search and can execute RDQL queries, including federation.

Most of the ontology development environments (as protégé, OntoStudio, NeOn Toolkit) have component -based architecture and can include many different visualization and querying tools. Some of the ontology development environments focus on the editing and management of ontologies, while others provide sophisticated features such as programmatically accessible interfaces, ontology visualization, querying, collaboration and usage of web services. Solutions, as KAON, WebOde , and Protégé provide ontology services. The main problem in these environments is related to the insufficient relatedness between different views that leads to the difficulties in its effective usage.

Domain-specific ontology – visualization and editing tools apart from general purpose visualizing and querying tools include tools for visualization of specific domain knowledge. For example, SMART (Semantic web information Management with automated Reasoning Tool) provide intuitive tools for life scientists for represent, integrate, manage and query heterogeneous and distributed biological knowledge. It uses AJAX, SVG and JSF technologies, RDF, OWL, SPARQL , DL reasoners (Pellet) for the automated reasoning.

Web – based Semantic Web tools are built in web portals or repositories and/or have searching and browsing capabilities. Web – based tools are mainly dedicated for ontology usage (searching, browsing, querying, information integration, etc.). Most tools are very useful for exploring the data offered by a single SPARQL endpoint, but their functionality for exploring the linked geospatial data cloud is very limited. The LODVisualization tool for example [9] is a very promising tool based on the Linked Data Visualization Model for visualizing RDF data.

OntoQuery utility [12], is an easy-to-use web-based OWL query tool with label replacement, syntax highlighting and checking and auto-complete.

QueryMed [13] is a distributed query engine in the biomedical domain that can execute queries relevant to a wide range of biomedical topics, runs federated queries across multiple SPARQL endpoints, and is designed to be usable by the users who do not know the structure of the underlying ontologies or the syntaxes of the SPARQL query language.

Rhizomer [19] is a semantic metadata editor and browser. For end-users, it proposes web page – based interface and semantics lay behind and is used to improve the user experience through an AJAX-enhanced web interface. It offers a SPARQL endpoint and wiki engine All these components are built in a powerful platform for Semantic Web portals.

Some of web based tools can visualize ontologies, serialized in very restricted number of formats (one or two). For example, OntoVisT is a web based ontological visualization tool, working with ontology files in OBO format. It is designed for interactive visualization of any ontological hierarchy, navigation of complex networks. It also can use search criteria, zoom in/out, center focus, nearest neighbor highlights and mouse hover events. There is also web – based ontology editors, supporting OWL format (web protege for example).

### 4.5 Visual Querying Tools

Although there are a lot of graphical visualization tools, most of its visual interfaces are usable for editing, but not for querying. The form – based querying interfaces rarely are linked by graphical tools and writing textual queries by selection of components in corresponding graphical view does not work. [17] present OntoVQL, a graphical query language for OWL-DL ontologies, but we couldn't find implementation of this language. Authors said it supports the user in developing syntactically valid queries, but the way of graphical representation of the query is difficult to understand by users, which are not familiar with logical knowledge representation.

The graphical visKWQL language [18] has the full expressivity of the underlying textual language, and use boxes for query elements, and relationship between them. It provides advantages mainly for inexperienced and intermediate users, as it should motivate those users to visually construct queries, preventing them from errors and display features of the system, helping them to create the queries they want. The main advantage of this visual language and query builder is that it is usable only with KWQL language.

NITELIGHT tool supports end users by providing a set of graphical notations that represent semantic query language constructs. This language provides a visual query language counterpart to SPARQL called vSPARQL. NITELIGHT also provides an interactive graphical editing environment that combines ontology navigation capabilities with graphical query visualization techniques. This paper describes the functionality and user interaction features of the NITELIGHT

tool based on our work to date. We also present details of the vSPARQL constructs used to support the graphical representation of SPARQL queries.

### 4.6 SPARQL Endpoints

SPARQL endpoints are web – based systems, allowing extraction of data, metadata or subontologies from semantic web repositories. Using SPARQL language and protocol, building in endpoints query execution engines analyze, execute queries and return results, extracted from one or more sources. SPARQL endpoints have two main component**s:** server component and client component. Server component include data store, software libraries and APIs. Client component include graphical interfaces and tools for easy query building. Software libraries and APIs have three main purposes: query analysis, query execution, and federation.

Many SPARQL endpoints use the ARQ library of the Jena Semantic Web Toolkit for parsing the query. Query patterns are parsed and mapped into axiom templates. Then they are passed to a query optimizer, which applies the axiom template rewriting and then searches for a good query execution plan based on statistics provided by the used reasoner. Returned results are presented according to query type, agent needs or visualization specifics.

NCBO for example is releasing a free and open SPARQL endpoint to query ontologies hosted in the BioPortal ontology repository. SPARQL federated queries are managed by Jena ARQ library. The Jena ARQ library handles the SERVICE SPARQL construct sets of triple patterns to different endpoint and handles the joins. SPARQL Federation can be used programmatically and is of great importance, because of Web of Data has no global schema, and querying data from multiple sources could be solved by link establishing among datasets. There are three federation architectural categories developed recently [6]: using SPARQL 1.1 Federation Extensions (ARQ a query engine, Sesame, SPARQL-FED, SPARQL-DQP), Frameworks build on top of SPARQL 1.0 and Frameworks build on top of SPARQL 1.1. Federation frameworks are far from maturity. The challenges that need to be tackled are related to Data Access and Security, Data Allocation and freshness, dealing with Overlapping Terminologies and provenance.

SPARQL endpoints client components usually are SPARQL web forms, Ajax based Visual Query Builders, Linked Data interfaces or simply web sites. [7] proposes a question-based Interface to Ontologies (QuestIO) - a tool for querying ontologies using unconstrained language-based queries

[4] has investigated enormous SPARQL infrastructure in the web, including at about 430 endpoints, and concludes that this infrastructure is not ready for action, because of:

Only one-third of endpoints make descriptive meta-data available, and it is difficult to locate or learn about the content and capabilities of others;

The support for established SPARQL features like ORDER BY and new SPARQL 1.1 features is realized only in small part of endpoints.

The performance of endpoints for generic queries can vary by up to 3–4 orders of magnitude.

This leads to the conclusion, that continuous monitoring of SPARQL endpoints is required.

### 4.7 Ontology Library Querying and Visualization Services

Users need visualization support for understanding of the used ontologies and machine accessible interfaces for using ontologies by querying or change information that is up to date. Implementation of the library interfaces for ontology usage can be based on: language based API's, HTTP communication by passing SOAP messages, REST-based approach, or AJAX-technologies. Ontology library systems should provide interfaces for both applications and human users for Searching, Browsing, Programmatic access, querying and other domain-specific tools(figure 5)

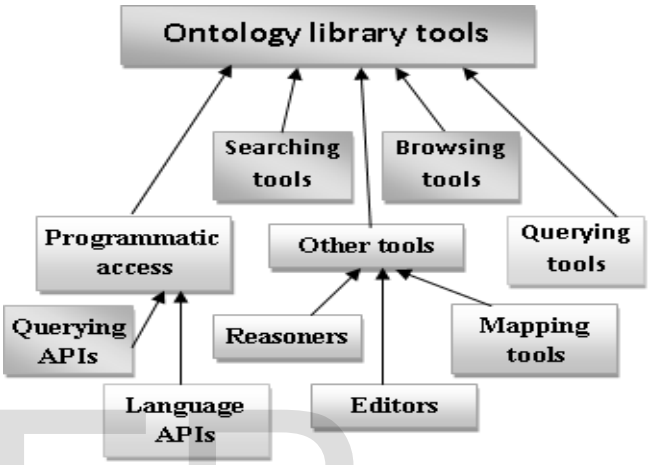



Fig. 5. Classification of **ontology library tools**

1. Searching
2. Simple keyword-based search mechanism over its collection of ontologies.
3. Ontology metadata based search
4. Downloading different versions of stored ontologies
5. Browsing:

All users, using an ontology library system require ontology visualization for gaining a clear perspective to the ontology as a whole. Browsers should provide navigation of the ontological relations, searching concepts by semantics or keywords. The browser should also provide different abstraction levels to the ontology according to the user group, because users have different knowledge about ontologies. Some ontology browsing tools are:

Ontology browsing interfaces as OWLSight

ONKI and Cupboard display a hyperlinked representation of the entities, which users can navigate.

OLS provides a tree representation of the ontology, which gives access to the complete description of each of the entities it contains.

Querying, based on the SPARQL language, reducing the set of ontologies into the DL-Lite formalism to enable the use of inferences in real time, during the querying process. Service for querying is required to enable interaction between an application and the ontology library system. Therefore a semantic web query language interface should be provided to an ontology library system.

Programmatic access

Through Web service protocols, such as SOAP and REST.

The Web services usually provide access to different types of content, including search functionalities, extracting metadata about the ontologies, searching through ontology content, and query access, such as a SPARQL endpoint.

Using programmatic access, Web services, for example can directly query the ARQ query engine for Jena which proposes flexible support of Standard SPARQL, free text search via Lucene, Access and extension of the SPARQL algebra, custom filter functions, Property functions, federated queries.

Other tools, as Versioning, Reasoning, mapping, User management and Notifications also may be included in ontology libraries. Such tools can collect valuable for searching metadata. Mappings and other inter ontology relations are usually included in ontology libraries. Specific tools, working with ontology metadata, are also used in many libraries. Such tools, for example, calculate a number of metrics on an every ontology when it is uploaded to the library ( Noy, & d'Aquin, 2011). The most widely used metrics include simple quantitative information such as the number of classes, properties, imported relationships between ontologies, the DL sublanguage that an OWL DL ontology falls into, number of classes with no documentation, authors who have contributed to the ontology, and so on. All these stored metadata about ontologies are of great importance in ontology searching process.

ONKI Ontology Service for example includes the FinnONTO infrastructure containing knowledge and web services in over than 400 domains, including the ONKI mash-up widgets. It also includes the notion of creating and maintaining a Linked Open Ontology Cloud KOKO that covers different domains, and is provided as a national centralized service [10].

Another approach to make Semantic Web Querying easy for all web users is usage of restricted natural language interfaces for querying. QuestIO system for example [15] aims to bring the simplicity of Google's search interface to conceptual retrieval by automatically converting short conceptual queries into formal ones, which can then be executed against any semantic repository. Another approach for developing domain-independent natural language interface for the Semantic Web is Querix [16], where natural language ambiguity problem are treated by asking the user for clarification in case of ambiguities. [20] propose ontology-anchored integrative query tool, Research-IQ, which employs a combination of conceptual knowledge engineering and information retrieval techniques to enable the intuitive construction of queries.

The systems, based on this approach are far from its maturity. The quality of returned results heavily depends on the quality and choice of vocabulary of the ontology, query clarity, user interaction.

## 5 The Integrated Ontology Development and Querying Architecture

Our brief analysis of linked data and ontology visualization and querying tools leads to several important conclusions:

1. There are a grand variety of such tools, and choosing the right one is not an easy tack;
2. Almost all tools require some competences in Semantic Web technologies, and are not usable for usual web users;
3. Almost all visualization tools have no capabilities to extract and show consistent subontologies;
4. Almost all querying tools are difficult to use by non professionals in ontology querying;
5. Integrated ontology development and querying environment doesn't ensure usage of all ontology languages, profiles;
6. Many different views may be integrated and used simultaneously in some IDEs as protégé, but they usually are not linked to each other;
7. To ensure information integration, querying of aligned ontologies, or federation querying tools are needed.

As there are some big integrated ontology development and querying environments but they are far from it maturity (no one include all needed tools for development and querying, needed for different level professionals, or for testing all views and tools, needed for non professionals), the new modular architecture for ontology building, evaluation and querying semantic web is needed. The main purpose of this architecture is to ensure easy usage of all needed for ontology development, testing (including by mapping and querying) and evolution tools from one and the same environment. It also should ensure working with different ontology formats, including usage of domain – specific knowledge representation tools.

As it is clear from our brief survey, ontology languages, ontology visualization and query technologies and tools are too much to be inserted as components of a single environment. Therefore it is essential that a integrated development and querying environment should be highly configurable and extensible. That is why we present the modular architecture,

We propose flexible modular and layered architecture that ensure easy dynamic import and integration only of these tools, needed for the current project. The main idea is that only the most frequently used tools should included by default, and there are a grand variety of tools, that user can include following the advice of a special tool called Configuration recommender. All the tools for ontology and linked data development, publication and querying should be described by usage of standardized vocabulary (as ontology). The recommendation system for discovering and choosing the right tool for specific tack (Configuration recommender) will use this ontology to recommend to the user the best tool for it needs in every moment.

The purpose of the layered architecture is to separate presentation, application logic, web data and extension software access from each other to enable a clear system design. Our architecture has three layers: visualization layer, development layer and web repository layer.

Ontology and linked data visualization layer includes:

1. Ontology hierarchy visualization window (including several views);
2. Textual query view window;
3. Form – based and/or menu-based query view window (different forms for different query types);
4. Graphical query results viewer;
5. Textual query results viewer;
6. Restricted natural language query interface window for non expert users;

7. Domain-specific visualization interfaces;

The main components of ontology and linked data API layer development layer) are:

8. Application programming interfaces (OWL API, Jena ARQ, etc.);
9. Reasoners;
10. Converting tools;
11. Ontology Query API for extracting ontology metadata, as language, logic, etc.;
12. Links to Web repositories;

Components, used in the two layers are:

13. Configuration recommender;
14. Tools ontology.

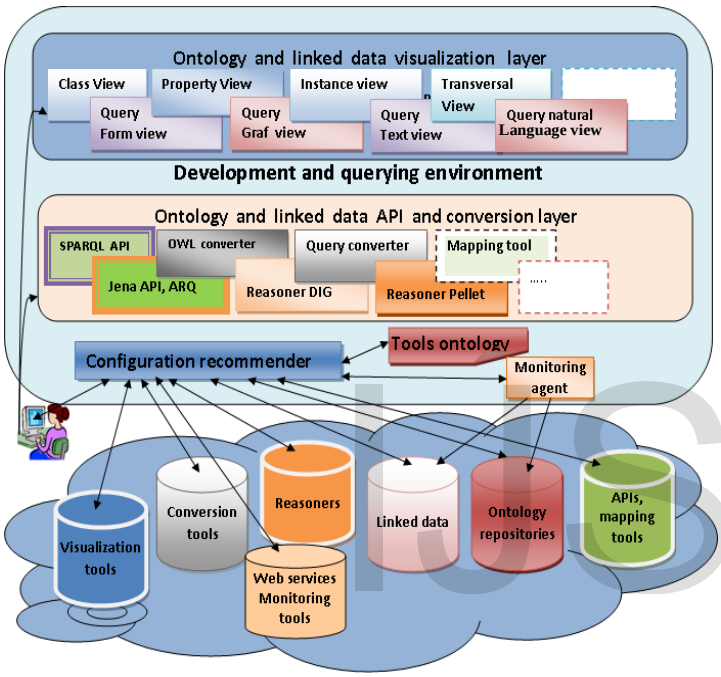



Fig. 6. The integrated ontology development and querying architecture

Web repository layer includes web-based linked data, knowledge, and web-based tools and services (including cloud- based ones), known and accessible from development and visualization layers. Some of these tools may be recommended by the configuration recommender and used in visualization layer, development layer, or in both layers. Web tools can be downloaded and included in the development layer, or directly used from the cloud.

Various APIs, reasoners and language converters (OWL API, Jena API, sesame and so on) are needed to be easily accessible for flexible support of usage of different ontology languages and profiles. All frequently used query languages and several query engines (as ARQ for Jena, Standard SPARQL, and extensions of the SPARQL algebra, custom filter functions, federated queries) also needed to be easily accessible for easy testing of various query responses.

When ontology developer modifies, or extends ontology, or proposes mapping between ontologies, it is important that he can easily predict how the ontology change will affect the typical query results, or representation in different type views. That is why it is important for developers to have all needed for query and ontology visualization tools in it development environment.

Before starting ontology development from scratch, knowledge engineers should search ontology libraries for similar ontologies that may be simplify the development process. Extraction of schema or instances from Semantic Web repositories by sending SPARQL queries also is useful as a variant of reusing knowledge. According to (Buil-Aranda, Hogan, Umbrich, & Vandenbussche, 2013), a grand number of SPARQL endpoints frequently are down, or don't support some complex query features. That is why SPARQL endpoints monitoring tool is needed to support Configuration recommender with the information about the current state of SPARQL endpoints. The main characteristics of public SPARQL endpoints are: Discoverability, Interoperability, efficiency, performance, availability, and support of SPARQL features like ORDER BY as well as for new SPARQL 1.1 features. Query converters and query mediators use them to decide how to transform a user query into several sub queries generates and integrates results from the distributed data sources, or to make some syntactic query conversion according to the specific syntactic requirements of the endpoints.

Sending typical queries is useful in the process of testing developed ontology or checking effects from mappings between two or more ontologies. Many different types of views are needed for making query process easy, simple and minimize possible errors. Class, property and instance views are useful when the query should related only to the schema or instances, focused and transversal views helps in abstracting from ontology components, not related to the tested sub ontology of large ontology. And high quality visualization (including coloring, 3D effects, drag and drop capabilities, etc.) is very useful when we work with big ontology. Query Graf view is very useful for visualizing results of specific type queries as construct queries. And different users have different preferences related to visualization techniques. Some professionals prefer to use form–based interfaces for query formulation, or write queries directly using SPARQL. And lay users are need for full automation in query writing (by graphical or menu-based interface, including search and filtering capabilities). Faceted or pivot –based views are useful when user should interact with massive amounts of data. That is who a grand number of visualization, form-based or text – based views should be accessible, customizable and easily importable in the environment.

Query natural language view is important mainly for lay users, but developers should in some cases test the quality of the mappings between restricted natural language and specific query language representation of the every type queries, expected for the ontology according to it representation language and underlined logic.

Tools ontology should describe all the tools, useful in the process of query making, sending, receiving or reasoning about the results. As there are a grand variety of tools, and each one can be described, using enormous amount of concepts and properties, one of the most difficult tacks will be to select only these tool characteristics that are important for querying Semantic Web. Axioms, stating what is better in various situations should also be included. All the knowledge

should be clearly directed to the strictly specified groups of users (having similar professional skills and working goals).

Various web – based tools may be recommended from modern cloud – based services as SMART tools, highly domain specific as Bioclipse for visualization and working with biomedical knowledge, to the most general tools as web protégé. The SPARQL Endpoints Status tool may be accessed in the internet. It monitors the availability, performance, interoperability and discoverability of SPARQL Endpoints registered in Datahub, and may help users in finding of needed semantic web sources. The system should include he's own monitoring tool for monitoring the activity of important web sites, or searching newly published data, knowledge or useful tools.

## 6 Conclusion

In this paper the brief analysis and classification of Semantic Web languages, technologies and tools is made and main conclusions are that heterogeneity in the semantic web required to be adequately managed by usage of many highly specialized tools. At the same time, Semantic Web complexity must be hidden in different ways and at different levels for different groups of users.

To ensure exploration, usage and evolution of Semantic Web Data and knowledge the flexible modular architecture for development, visualization and querying Semantic Web is proposed. Our architecture is in some way similar to the other component – based architectures (as Eclipse for example), but it differ in usage of Semantic Web technologies for tool description and the higher level of automation in tool selection. Our future plans are to develop the prototype of the Tools ontology and Configuration recommender, and use protégé and protégé plug-ins to make more flexible and easily configurable semantic web development, querying and visualization environment.

## References

[1] P. Ziegler C., Sturm, and K. R. Dittrich, "Unified Querying of Ontology Languages with the SIRUP OntologyQuery API". In BTW pp. 325-344, 2005.

[2] E. Sirin and B. Parsia, "SPARQL-DL: SPARQL Query for OWL-DL", *Proc. 3rd OWL Experiences and Directions Workshop*., 2007.

[3] R. Fikes, P. Hayes, and I. Horrocks, "OWL-QL—a language for deductive query answering on the Semantic Web.", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 2, no. 1, pp. 19-29, 2004.

[4] C. Buil-Aranda, A. Hogan, J. Umbrich, and P. Y. Vandenbussche, "SPARQL Web-Querying Infrastructure: Ready for Action?", Proc. *Semantic Web–ISWC,* pp. 277-293, 2013, Springer Berlin Heidelberg.

[5] N. F. Noy, and M. A. Musen, " Specifying ontology views by traversal. Proc. *Semantic Web–ISWC,* pp. 713-725, 2004.

[6] N. A. Rakhmawati et al., "Querying over Federated SPARQL Endpoints---A State of the Art Survey.", DERI Technical Report arXiv preprint arXiv:1306.1723,2013.

[7] D.Damljanovic, V. Tablan, and K. Bontcheva, "A Text-based Query Interface to OWL Ontologies.", *In LREC.* , 2008.

[8] K. Bereta et al., "SexTant: Visualizing Time-Evolving Linked Geospatial Data.", Proc. *International Semantic Web Conference,* pp. 177-180, 2013.

[9] J.M.B. Ferneandez, S. Auer, R. Garcia, "The linked data visualization model.", Proc. *International Semantic Web Conference (Posters & Demos),* 2012.

[10] E. Hyvönen, J. Tuominen, M. Alonen and E. Mäkelä, "Linked Data Finland: A 7-star Model and Platform for Publishing and Re-using Linked Datasets." Proc. *ESWC,* Demo and Poster Papers, 2014.

[11] N. F.Noy and M. d'Aquin, "Where to publish and find ontologies? A survey of ontology libraries.", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 11, 2011.

[12] I.Tudose et al., "OntoQuery: easy-to-use web-based OWL querying.", *Bioinformatics*, vol. 29 no 22, pp. 2955-2957, 2013.

[13] O. Seneviratne, and R. Sealfon, "QueryMed: An Intuitive Federated SPARQL Query Builder for Biomedical RDF Data." http://querymed.googlecode.com/svn-history/r68/trunk/docs/www2010/paper.pdf, 2010.

[14] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou, "Ontology visualization methods—a survey.", *ACM Computing Surveys (CSUR),* vol. 39, no. 4, 2007.

[15] V. Tablan, D. Damljanovic and K. Bontcheva, "A natural language query interface to structured information", *Springer Berlin Heidelberg*. pp. 361-375, 2008.

[16] E. Kaufmann, A. Bernstein and R. Zumstein, "Querix: A natural language interface to query ontologies based on clarification dialogs." Proc. *5th International Semantic Web Conference (ISWC)*, pp. 980-981, 2006.

[17] T. Catarci, M. F. Costabile, S. Levialdi, and C. Batini, "Visual Query Systems for Databases: A Survey.", *Journal of Visual Languages and Computing* vol. 8, no. 2, pp. 215-260, 1997.

[18] A. Hartl, "A Visual Rendering of a Semantic Wiki Query Language", PhD dissertation, Institute of Computer Science, LMU, Munich, 2009.

[19] J. M. Brunetti, R. García and S. Auer, "From Overview to Facets and Pivoting for Interactive Exploration of Semantic Web Data.", *International Journal on Semantic Web and Information Systems (IJSWIS),* vol. 9, no. 1, pp. 1-20, 2013.

[20] T. B. Borlawsky, , Lele, O., & Payne, P. R. "Research-IQ: development and evaluation of an ontology-anchored integrative query tool." *Journal of biomedical informatics*, vol. 44, pp.56-62, 2011.