Evaluating Semantic Web: Creating Semantically Enriched Web Pages for a Medical Use Case

Ioannis Gkourtzounis
Department of Computing, The University of Northampton,
Park Campus, Boughton Green Road, NN2 7AL, Northampton, UK
ioannisgk@live.com

***Abstract*—The evolution of the web with web applications, web services and the use of social networks, created a new need to automate complicated tasks. Software agents need to understand and interpret data, and Semantic Web gives us the techniques to accomplish that, where efficient knowledge reuse among users and agents is the ultimate goal. In this paper we explore the theoretical background of Semantic Web, presenting different data representation standards like XML, RDF, RDFS and OWL. Ontologies are the backbone of Semantic Web technologies, which are used extensively in the health domain. We have analyzed, designed and developed a solution for a medical use case where a user can extract information in form of Microdata, related to Medical Conditions. Our web application, Microdata WebParser, can also convert Microdata to other popular formats, like RDFa and JSON-LD. Taking into account the theoretical exploration and our implementation of the application, we argue that (i) creating new semantically enriched web pages by combining data from different sources can be achieved with Semantic Web techniques and (ii) some of the most important challenges of Semantic Web are the visibility of data repositories and the outdated and inconsistent datasets.**

***Keywords*—Semantic web, knowledge representation, resource description framework, ontology web language, artificial intelligence, software agents.**

I. Introduction

Today the web is a vast pool of information that is supported by the world's network infrastructures and delivers data to its users. Web pages present data to humans and they succeed at what they do. But what happens when we want data to be readable by software that automate specific processes? How can a program "understand" and interpret those data? The language of the web is HTML, where tags are used to describe the format of a web page. However, those tags do not reveal the semantic meaning of the page itself or its contents [1]. Humans can easily figure out their meaning, for example if a number on a page represents a price. But semantic ambiguity is a serious problem for a software that needs to make decisions based on data. No universal standard for expressing the meaning of information leads to limited knowledge reuse, and that is a problem [2].

The idea of Semantic Web came from W3C director Tim Berners-Lee's vision that emphasized on knowledge exchange [3]. It is an extension of the web, that provides a common framework for humans and machines to understand and use data effectively [4, 5]. In this paper we examine Semantic Web in more depth, how data are represented and what are the main challenges today. The third section focuses on the medical applications of Semantic Web and a proposed solution in a specific use case about Medical Conditions. In the last section we will point out our findings and evaluate Semantic Web techniques to create semantically enriched web pages.

II. Background

A. Semantic Web

The evolution of the Internet has been remarkable, allowing programmers and users accomplish more, with less effort at every evolution cycle (eg. Web 1.0, 2.0, 3.0). Internet allowed the development of programs that could communicate, without the programmer to worry about the hardware network infrastructure. The web made possible for users to work with interconnected documents, without worrying about the computers that store them. Semantic Web is the third level of abstraction that allows programmers and users to reference real world objects without concerning with how they are described [6]. The main objective of Semantic Web is to provide structure to the meaningful content of web pages, so that software agents[1] can execute complicated tasks [7]. For example, an agent to a clinic website will not only know the content keywords, but also which days a specific doctor works so it can choose appointment times by providing possible dates to the page's script.

The Semantic Web can be considered as a large online database, with structured information that can be queried. The difference with traditional databases is that information can be heterogeneous, contradictory and incomplete [2]. When structured collections of information and inference rules that conduct automated reasoning are available, knowledge can be represented and exchanged between users and agents [7]. And that, is the power of Semantic Web. Terms like automated decisions, inference and description logic were already used in Artificial Intelligence and that created confusion, but we need to keep in mind that Semantic Web is more about knowledge representation and less about reasoning [8].

B. Data Representation

Data can be represented in different ways using different techniques. One way is to create tags and allow users to define

[1] An agent is a piece of software working autonomously and continuously in a particular environment in order to achieve a specific purpose.

Fig. 1. Most used representation models for Semantic Web.

their content using XML [5]. These tags include attribute-value pairs and they give structure to the document, without using a fixed vocabulary [9]. The Resource Description Framework (RDF) on the other hand, has three types of elements as its building blocks: (i) the resources, identified by Uniform Resource Identifiers (URIs), (ii) literals that are atomic values like strings and numbers, and (iii) properties which are binary relationships between resources and literals [5, 10]. A relationship between two resources or between a resource and a literal is called a triple [9, 11]. The triple is an important advantage over XML where only attribute-value pairs are used. RDF can be regarded as an alphabet that allows a system to build words and sentences but not a language. Meaning is given when using a vocabulary that defines the meaning of RDF statements. RDF Schema (RDFS) is such a vocabulary that incorporates class membership, sub-class hierarchies, class attributes and sub-property hierarchies [2, 9]. This way, RDFS encodes the meaning of documents in sets of triples using a vocabulary.

An ontology is a collection of information with a taxonomy that defines the classes and relationships, and a set of inference rules that empowers them with reasoning [7]. The Web Ontology Language (OWL) is based on these formal conceptualizations of particular domains. OWL follows the requirements and design principles from a specific field and expresses more meaning and semantics than XML, RDF and RDFS [12]. Fig. 1 shows the representation models of Semantic Web and on the top layer some of the most used vocabularies that are defined using RDFS and OWL: FOAF (friend-of-a-friend), SKOS (simple knowledge organization system) and COMM (core ontology for multimedia).

Other standards of Semantic Web include Linked Data, where an application or agent can start at one piece of Linked Data and follow embedded links to other pieces on different websites [13]. Microdata annotate information in documents with specific machine readable labels, making them more visible to search engines [14]. Data are recognized as a set of items described by property name and value pairs. JSON-LD is a JSON based format used to serialize Linked Data when building web services. JSON-LD follows a strict RDF syntax, so a JSON-LD document also represents an instance of an RDF data model [13]. The most widely used standard to query knowledge in Semantic Web, is the SPARQL language. We can interact with SPARQL endpoint implementations and extract results in specified formats [12].

C. Main Challenges

Although the Semantic Web gives us valuable tools for automated software to interpret data and perform sophisticated tasks, it faces some important challenges. First of all, it is difficult to create a ”universal ontology for everything” due to huge space complexity, so distributed ontology development is preferred [15]. This introduces even more problems, like ontology integration, mapping, translation, consistency and aging, where data from old ontologies can not be used in a meaningful way [16].

Furthermore, low quality datasets with format problems generate query issues and only add to the slow worldwide adoption of Semantic Web technologies [17]. Issues in heterogeneous data, the mismatch of data models and immature best practises, make the life of the programmers even more difficult when implementing Semantic Web applications [18]. Searching for data with SPARQL queries, require specific knowledge of the underlying data, so retrieval of information remains a difficult task as many data remain hidden [19].

III. Medical Applications

The power of Semantic Web is revealed when programs collect data from different sources, process them and exchange the results with other programs. Such software agents will have increased effectiveness as more readable content and automated services become available [7]. Databases that generate new information by creating new and updated ontologies, bring important benefits to research communities where complex knowledge is available, like in clinical and biomedical fields [10]. Indeed, Semantic Web technology is used extensively in the health domain [20] and opens new opportunities in health care management where agents perform intelligent tasks for the users [21]. Semantic web services seem very promising in medical health planning and generally in any field where human and agents interact semantically with the data [22].

A. Problem Statement

Focusing on the medical applications of Semantic Web, we will apply the techniques we discussed earlier on our medical use case. We assume that a client requests a program to extract information related to Medical Conditions from different sources, combine them and generate a new semantically enriched web page. The source data should be in Microdata format and the application should also convert them to RDFa[2] and JSON-LD compliant files. The client will be able to select a Medical Condition from a list and select an action. The appropriate sources will be parsed, analyzed, the Microdata will be extracted and a new web page will be created.

[2]RDFa is a syntax that allows us to embed RDF information into HTML documents via attributes.

B. Analysis & Design

In order to build the software, we first had to conduct a research on Microdata websites related to Medical Conditions. So, the first step is to find search engines that search in websites that follow the guidelines of Microdata. Such a website, should have the appropriate tags as per Schema.org, a popular structured data markup schema for Microdata. The candidate websites have to incorporate the properties and values as described in the "MedicalCondition" entity in health-lifesci.schema.org.

The requirements for the web application are the following:

- The user can select a Medical Condition from a list
- The software should load related sources, and also:
- Parse sources for Microdata attributes and values
- Process the data (for example concatenate text for the same attributes)
- Generate a web page with the new data, combining attributes and values from sources, in Microdata format
- Convert Microdata to RDFa format
- Convert Microdata to JSON-LD format

The first problem that we encountered was finding the suitable sources for our use case. We examined carefully the most popular Semantic data repositories related to Medical Conditions [23] to see if these websites use the Microdata format. The Global Health Observatory[3] offers datasets in HTML, CSV, XML, JSON and Excel formats, but not in Microdata. Bio2RDF[4] is a biological knowledge base that contains RDF data and Microdata, but they are not grouped in Medical Conditions categories. CardioSHARE[5] only incorporates a SPARQL query engine for RDF data and this was not helpful in our case. So, the next step was to find a source code search engine. Our search in PublicWWW[6] for the term "http://schema.org/MedicalCondition" gave us our main sources with Microdata: blausen.com and coreem.net.

For the development of our web application we chose the Java Spring Framework, a well established application framework, fully compatible with the Model-View-Controller (MVC) pattern. It has a layered architecture and uses Services classes to process data and pass them to Models. The data are sent to the Controllers that make them available to specific JSP pages. Web browsers can interpret the JSP pages just like simple HTML pages. Spring Framework has formalized the best practises as design patterns and it is widely used for developing GUI applications, web applications and applets.

The graphical user interface consists of three main pages. On the first page, the user is presented with a list of Medical Conditions where he can select one condition and click on "Generate" to move on to the second page. In the background, the application loads the related sources, parses them and recognizes the properties and values from the Microdata on the source web pages. On the second page, the sources, properties and values are shown in tables and the user has the following options: generate Microdata, generate RDFa code or generate JSON-LD code. The third page processes the data according to the user's selection and generates the corresponding code (Fig. 2).

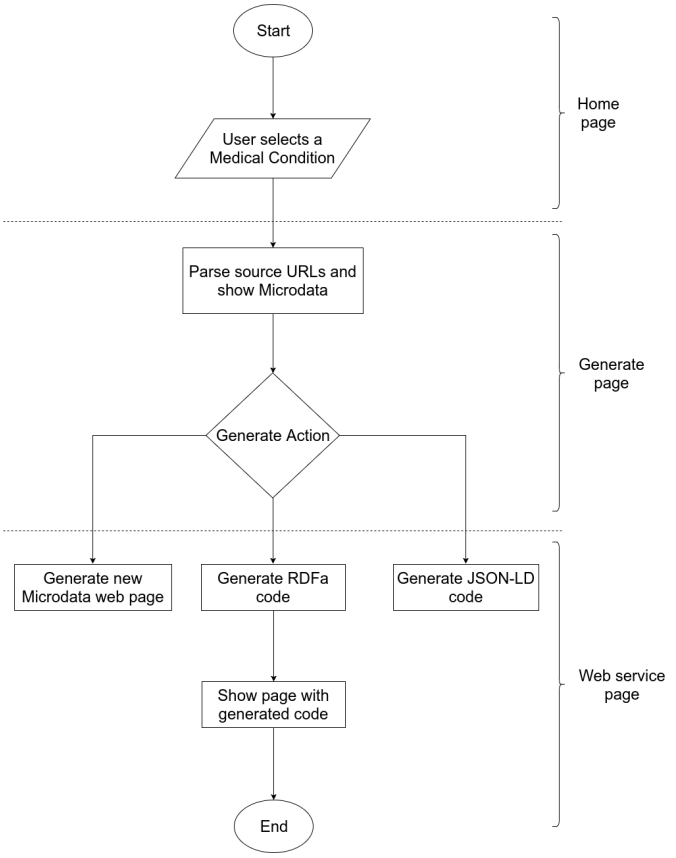

Fig. 2. Microdata WebParser flow diagram. The user selects a Medical Condition, the application parses the source URLs and then generates the desired code.

For developing the application according to Spring and MVC standards, we needed a HomeController class, responsible for getting the sources, calling Services classes, adding data to the Model and showing specific JSP pages. The GetSourcesService class gets all sources as a HashMap from SourcesDAO where the URLs are stored. After studying the source code from our sources, we decided to create two separate Services for parsing the web pages because each website used a slight different approach on incorporating Microdata (for example, adding empty values on certain descriptions). ParseMode1Service and ParseMode2Service classes serve this purpose. Finally, GenerateMicrodataService, GenerateRDFaService and GenerateJSONLDService, provide the functionality for creating a new web page with combined Microdata and generating RDFa or JSON-LD code (Fig. 3).

C. Implementation

Following the best practises of Spring Framework, we organized our code into three packages. The controller package includes the HomeController class that uses instances of the

[3]http://apps.who.int/gho/data/node.home
[4]http://bio2rdf.org/
[5]http://biordf.net/cardioSHARE/
[6]https://publicwww.com/

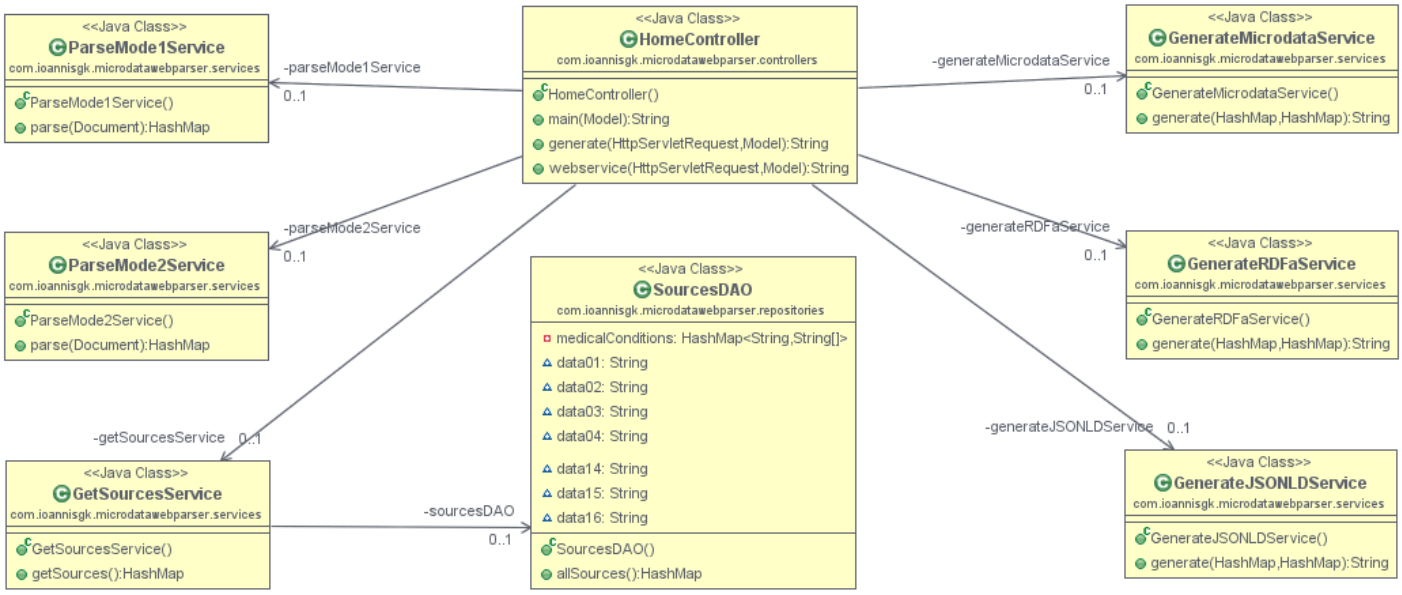

Fig. 3. Class diagram of our web application that extracts Microdata, processes them and creates a new semantically enriched web page.

Services and their methods to process data and call the corresponding JSP pages. The repositories package contains the SourceDAO class. The Medical Conditions are represented as a HashMap with String keys and array of String values. The constructor of the class initializes each key with the URLs that contain Microdata related to a Medical Condition. The GetSourcesService in the Services package provides the method getSources to the HomeController.

We used the Jsoup library for parsing the HTML content and recognizing the ”itemscope”, ”itemprop” and ”itemtype” tags which were present in our sources. In ParseMode1Service we selected the document, iterated the elements and discarded the variable containing certain values (like the ”View disclaimer” and ”...” characters). This was done because those values carried semantically incorrect information. SchemaKeywords and SchemaDescription are ArrayLists with the actual Microdata and we saved them in the extractedCondition HashMap so they could be used later (see Appendix A). Following the same logic, we implemented ParseMode2Service while recognizing treatment and pathophysiology as extra fields in the Microdata from the second source.

When generating a new web page with Microdata, we populated the required HTML code with the ”itemscope”, ”itemprop” and ”itemtype” tags. The variables included in the new Microdata web page are: name, medicalSpecialty and description from source 1 and alternateName, medicalSpecialty, pathophysiology and possibleTreatment from source 2. This way, we combined information from both sources and created a new semantically enriched web page (see Appendix A). The variable ”htmlCode” is passed from the HomeController to the Model and then to the JSP page that shows the semantic information in the form of two tables. By studying the structure and characteristics of RDFa and JSON-LD standards, we created the implementation of GenerateRDFaService and GenerateJSONLDService classes.

In our HomeController class we declare the HashMap variables extractedCondition1 and extractedCondition2 to hold the semantic data from our sources. User’s selection is stored in the String ”selection” and Strings ”source1” and ”source2” contain the URLs of the sources that are related to the Medical Condition selected. The semantic data are extracted and saved to extractedCondition1 and extractedCondition2 HashMaps by the Service. The type of code to be generated is retrieved from the String ”type”, so we call the ”generate” method from the appropriate Service to create the code we need. The String ”htmlCode” is then passed to the Model and used in the ”webservice” JSP page (see Appendix A).

Finally, we installed MySql and Tomcat 9.0 on a hosting server. Tomcat is a web server and servlet container from Apache that allows JSP and Java Servlets to run. We exported the WAR[7] file from Eclipse IDE and uploaded it to our server. We did not encountered any problems in the implementation, apart from some research and parsing challenges mentioned earlier. All source code is freely available on GitHub[8] and the final web application demo is live on the link: http://178.62.121.237:8080/microdata-webparser/home/main.

D. Testing

The next step was to perform black-box and white-box testing. In black-box testing, we did not consider the internal structure and we tested the software with different inputs and examined its behavior. We focused mainly on functional testing. We also noticed that our web application, Microdata WebParser, is a simple application with only eight options for user input. Each option gives three different outcomes, so

[7]A WAR file is a compressed package containing Java web applications.
[8]Full source code on GitHub: https://goo.gl/y8xRdX

Select	Generate	Expected Result	Status
Condition 01	Microdata	Show Microdata	PASS
Condition 01	RDFa	Show RDFa	PASS
Condition 01	JSON-LD	Show JSON-LD	PASS
Condition 02	Microdata	Show Microdata	PASS
Condition 02	RDFa	Show RDFa	PASS
Condition 02	JSON-LD	Show JSON-LD	PASS
Condition 03	Microdata	Show Microdata	PASS
Condition 03	RDFa	Show RDFa	PASS
Condition 03	JSON-LD	Show JSON-LD	PASS
Condition 04	Microdata	Show Microdata	PASS
Condition 04	RDFa	Show RDFa	PASS
Condition 04	JSON-LD	Show JSON-LD	PASS
Condition 05	Microdata	Show Microdata	PASS
Condition 05	RDFa	Show RDFa	PASS
Condition 05	JSON-LD	Show JSON-LD	PASS
Condition 06	Microdata	Show Microdata	PASS
Condition 06	RDFa	Show RDFa	PASS
Condition 06	JSON-LD	Show JSON-LD	PASS
Condition 07	Microdata	Show Microdata	PASS
Condition 07	RDFa	Show RDFa	PASS
Condition 07	JSON-LD	Show JSON-LD	PASS
Condition 08	Microdata	Show Microdata	PASS
Condition 08	RDFa	Show RDFa	PASS
Condition 08	JSON-LD	Show JSON-LD	PASS

Fig. 4. All test cases for our application with user selection, generate action, expected result and status, which is based on the actual and expected results.

we created 24 test cases in order to achieve 100% decision coverage for our code. This was very convenient and with those 24 test cases we applied both black-box and white-box test design techniques, as decision coverage deals with the internal structure of the software.

The table in Fig. 4 presents all test cases for our application. The first column shows the selected Medical Condition, the second column shows the action of the user, the expected result is on the third column and on the fourth column the Status indicates if the actual result is the same with the expected result. As we can see, all test cases passed the testing. The list of Medical Conditions used in the test cases are:

Condition 01: Abdominal Aneurysm
Condition 02: Asthma
Condition 03: Atrial Fibrillation
Condition 04: Ectopic Pregnancy
Condition 05: Herpes Zoster
Condition 06: Myocardial Infarction
Condition 07: Otitis Media
Condition 08: Pulmonary Embolism

IV. Conclusions

With its roots in Artificial Intelligence, Semantic Web focuses on knowledge representation with the main objective of efficient knowledge reuse. Creating new and updated ontologies from existing data, brings important benefits to research communities worldwide, like in biomedical fields

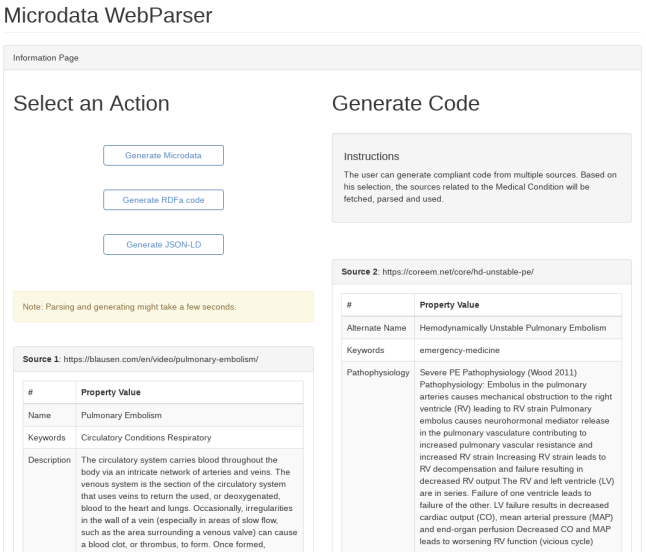

Fig. 5. The page where Microdata WebParser recognizes and displays Microdata from the sources. The user can generate a new page or convert Microdata to RDFa and JSON-LD formats.

and the health domain in general. However, Semantic Web faces some very important challenges. There is no universal ontology of everything and distributed ontology development introduces the problems of ontology integration, mapping, translation, consistency and aging. Low quality of datasets with format problems, the mismatch of data models and immature best practises generate query issues and make implementing Semantic Web applications a difficult job.

We analyzed, designed and developed a solution, Microdata WebParser (Fig. 5, more screenshots available in Appendix B), for a medical use case where a program should extract information in form of Microdata related to Medical Conditions and generate a new semantically enriched web page, RDFa and JSON-LD code. We encountered two important problems during the development. First, it was difficult to find suitable sources as even the most popular Semantic Web data repositories did not include web pages with Microdata. We turned our attention to source code search engines to find suitable sources. The second problem was that our sources used a slight different approach on incorporating Microdata in their code with some inconsistencies and format errors, so we had to create separate classes for parsing their source code.

Both the theoretical exploration and the implementation of our application, bring us to the conclusion that (i) creating new semantically enriched web pages by combining data from different sources can be achieved with Semantic Web techniques and (ii) some of the most important challenges of Semantic Web are the visibility of data repositories and the outdated and inconsistent datasets. Dealing with those challenges will reveal the true power of Semantic Web, and enhance knowledge reuse and exchange between users and agents.

REFERENCES

[1] P. Warren and J. Davies, "The Semantic Web From Vision to Reality", *ICT Futures*, pp. 53-66, 2008.

[2] E. Oren and S. Schenk, "Semantic Web Basics", *Multimedia Semantics*, pp. 81-98, 2011.

[3] O. Mustapaa, A. Karahoca, D. Karahoca and H. Uzunboylu, "Hello World, Web Mining for E-Learning", 2017. .

[4] D. Jeon and W. Kim, "Development of Semantic Decision Tree", 2011.

[5] Q. Quboa and M. Saraee, "A State-of-the-Art Survey on Semantic Web Mining", *Intelligent Information Management*, vol. 05, no. 01, pp. 10-17, 2013.

[6] J. Hendler and T. Berners-Lee, "From the Semantic Web to social machines: A research challenge for AI on the World Wide Web", *Artificial Intelligence*, vol. 174, no. 2, pp. 156-161, 2010.

[7] T. Berners-Lee, J. Hendler and O. Lassila, "The Semantic Web", *Scientific American*, vol. 284, no. 5, pp.28-37, 2001.

[8] "Semantic Web Misconceptions - Cambridge Semantics", *Cambridge Semantics*, 2017. [Online]. Available: https://www.cambridgesemantics.com/blog/semantic-university/intro-semantic-web/semantic-web-misconceptions/. [Accessed: 12- Dec- 2017].

[9] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann and I. Horrocks, "The Semantic Web: the roles of XML and RDF", *IEEE Internet Computing*, vol. 4, no. 5, pp. 63-73, 2000.

[10] V. Nebot and R. Berlanga, "Finding association rules in semantic web data", *Knowledge-Based Systems*, vol. 25, no. 1, pp. 51-62, 2012.

[11] "RDF 1.1 Concepts and Abstract Syntax", *W3.org*, 2017. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/. [Accessed: 12- Dec- 2017].

[12] E. Oren and S. Schenk, "Semantic Web Languages", *Multimedia Semantics*, pp. 99-128, 2011.

[13] "JSON-LD 1.0", *W3.org*, 2017. [Online]. Available: https://www.w3.org/TR/json-ld/. [Accessed: 12- Dec- 2017].

[14] A. Nogales, M. Sicilia, S. Snchez-Alonso and E. Garcia-Barriocanal, "Linking from Schema.org microdata to the Web of Linked Data: An empirical assessment", *Computer Standards Interfaces*, vol. 45, pp. 90-99, 2016.

[15] L. Ding, P. Kolari, Z. Ding and S. Avancha, "Using Ontologies in the Semantic Web: A Survey", *Integrated Series in Information Systems*, pp. 79-113, 2005.

[16] A. Tjo, A. Andjomshoaa, F. Shayeganfar and R. Wagner, "Semantic Web Challenges and New Requirements", *Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA05)*, 2005.

[17] J. Martinez-Rodriguez, I. Lopez-Arevalo and A. B. Rios-Alvarado, "A classification of challenges in the Semantic Web based on the general architecture", *26th International Workshop on Database and Expert Systems Applications*, 2015.

[18] B. Heitmann, S. Kinsella, C. Hayes and S. Decker, "Implementing Semantic Web applications: Reference Architecture and Challenges", *SWESE 2009: 5th International Workshop on Semantic Web Enabled Software Engineering*, 2009.

[19] X. Zenuni, B. Raufi, F. Ismaili and J. Ajdari, "State of the Art of Semantic Web for Healthcare", *Procedia - Social and Behavioral Sciences*, vol. 195, pp. 1990-1998, 2015.

[20] W. Woensel, N. Haider, P. Roy, A. Ahmad and S. Abidi, "A Comparison of Mobile Rule Engines for Reasoning on Semantic Web Based Health Data", *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, 2014.

[21] J. Niekerk and K. Griffiths, "Advancing Health Care Management with the Semantic Web", *2008 Third International Conference on Broadband Communications, Information Technology Biomedical Applications*, 2008.

[22] M. Gangwar, R. Yadav and R. Mishra, "Semantic Web Services for Medical Health Planning", *1st Intl Conf. on Recent Advances in Information Technology RAIT-2012*, 2012.

[23] X. Zenuni, B. Raufi, F. Ismaili and J. Ajdari, "State of the Art of Semantic Web for Healthcare", *Procedia - Social and Behavioral Sciences*, vol. 195, pp. 1990-1998, 2015.