Where Are All the Semantic Web Agents: Establishing Links Between Agent and Linked Data Web through Environment Abstraction

Oguz Dikenelli

Ege University, Department of Computer Engineering,
35100 Bornova, Izmir, Turkey
oguz.dikenelli@ege.edu.tr

Abstract. Semantic Web Agents have been considered as main software artifacts to consume the semantic data since the Semantic Web concept was raised first time in the well known "The Semantic Web" article in 2001. More then a decade passed and there is no collaboration between multi-agent systems and semantic web (or its current realization: the Linked Data Web) communities that can be considered important. In this paper, it is argued that initial vision was right and two comminities need each other to scale up their current practice. Thus, a conceptual framework is proposed to establish necessary links between agent and linked data web infrastructures. Environment abstraction has a special role in this framework and this role is especially discussed throughout the paper.

1 Introduction

In the beginning of the Semantic Web movement, agents have been considered as a first-class abstractions. The first article that put the Semantic Web concept forward is the well known "The Semantic Web" article written by Berners-Lee, Hendler, and Lassila [4]. The first two paragrafs of the paper describes a Semantic Web scenario where Lucy tries to schedule a series of physical therapy sessions for her mom together with her brother Pete. The second paragraph of the paper is rewritten below to emphasize the importance of Semantic Web Agent concept in the initial vision.

"At the doctor's office, Lucy instructed her Semantic Web agent through her handheld Web browser. The agent promptly retrieved information about Mom's prescribed treatment from the doctor's agent, looked up several lists of providers, and checked for the ones in-plan for Mom's insurance within a 20-mile radius of her home and with a rating of excellent or very good on trusted rating services. It then began trying to find a match between available appointment times (supplied by the agents of individual providers through their Web sites) and Pete's and Lucy's busy schedules. (The emphasized keywords indicate terms whose semantics, or meaning, were defined for the agent through the Semantic Web.)" As it is seen, (Semantic Web) agents are playing the leading role in the scenario.

Six years later, James Hendler wrote a letter as IEEE Intellegent Systems editor in May-June 2007 issue and asked an important question "Where are all the Intelligent Agents". He commented in the letter that key obstacles to the wider deployment of agents were identified early on as the needs for interoperability and intercommunication. He also argued that well established web services standards (and vendor(s) support) provide necessary interoperability infrastructure and Semantic Web standards provide knowledge sharing infrastructure for intelligent agent's intercommunications.

In 2007 when Hendler wrote the letter, RDF, RDF Schema and OWL had become standards. There was huge research effort on many areas such as new standard development (such as SPARQL), inference optimization, new ontology constructions for diffrent domains, ontology mapping and alligment languages and so on. Also, there was huge tool development effort which resulted some great open source tools to create and manipulate ontologies such as Protoge, Pellet, Jena and many others.

Despite of huge research efforts in mid 2000's, it was very doubtful to argue that there was a Semantic Web in that time as envisioned. Because, The Semantic Web was defined as a universal knowledge graph from the beginning where every concept named by a URI and these concepts are progressively linked into a universal web [4]. In 2007, there was lots of necessary standards, tools, and independent information systems that use these standards and tools but not a universal linked knowledge web. So the question would be "Where is the Semantic Web We Envisioned" for that time.

But, today it is certain that we are living in the era of transformation of web into a knowledge web which is also called Linked Data Web. There were 295 linked data sets in the Linked Data Cloud in August 2011 (the time when last snapshot was taken) creating more than 31 billion triples and 500 million out links [5]. Linked data web is contantly and exponentially growing which incudes datasets in many domains such as goverment, media, life sciences, geocraphic and etc.

So it is time for agent researchers and practitioners to ask themself how they can link agents to this new web. In this paper, a conceptual framewok is proposed to identify the linkage between the semantic agents and the linked data web.

2 Architectural Patterns of Linked Data Web

In July of 2006, Tim Berners-Lee published a personal note titled "Linked Data - Design Issues" [3]. Note was beginning with the following sentences; "The Semantic Web isn't just about putting data on the web. It is about making links, so that a person or machine can explore the web of data." I believe that this note is crucial for the Semantic Web evolution. Because, it changed the focus of the whole community to real problem; creating a web of data by making links. At that time, community was considering the ontologies like silver bullet and putting them into almost any information system problem known. There was many projects which use ontologies and related tools and many proposed do-

main ontologies, but creating a web of data was not a common vision within the community. Tim Berners-Lee reminded the community the original vision and renamed this vision as Linked Data Web to get rid of the confusion within the community.

After the publishment of "Linked Data - Design Issues", community have taken the message and began to create linked data web. Actually, there were only 4 principles in "Linked Data - Design Issues" note:

1. Use URIs as names for things.
2. Use HTTP URIs, so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
4. Include links to other URIs, so that they can discover more things.

By applying these principles, the community has created the Linked Data Cloud which included more than 31 billion triples and 500 million out links in August 2011 (last measurement, but constantly expanding). Today, we have a well established infrasturacture to publish and consume the linked data from the cloud.

To publish data, there are various ready-to-use data wrappers and converters for all widely used structural and semi-structural data models. Since most of the structural data on the Web are stored by relational databases, the most mature wrappers are for transforming relational databases structures to RDF model. W3C published a recommendation in September 2012 for a standard language called R2RML [8] to express customized mappings from relational databases to RDF datasets. R2RML has been widely accepted by tool developers, even well known open source D2R [7] uses R2RML as the mapping language.

There are, of course, many tools to convert application specific formats such as CVS, Excel, XML ext. into RDF. These tools are known as RDFizing tools and a list of them can be found in [6]. The output of these tools can be static RDF files accessed through a web server or converted RDF data can be loaded to an RDF Store.

Once the internal data sources are converted to RDF and/or necessary mappings are defined through wrappers to create RDF view, these created RDF data can be accessed via Web by two ways: querying with SPARQL protocol through SPARQL endpoints (RDBtoRDF wrappers and RDF stores provides endpoint support) or accessing RDF files (by dereferencing URL adresss) through web server.

Now consider that there hundreds (thousands in very near future) data sources publishing their data, there are many links (billions of them) between published sources and there is very dynamic environment where new sources enter and new links are created constantly and in an uncontrolled manner. Although the tools and approaches for publishing RDF data are stable and deployed succesfully in hundres of data sources in LOD cloud, consuming the desired data from such a highly dynamic environment is very challenging task. So, linked data community's main focus is to find effective ways to consume data from the Linked Data Cloud. This focus is also very critical for semantic

agent researchers since agents have been considered as the main consumer as it is discussed in the introduction.

There are three well undestood architectural patterns for consuming linked data depending on the application requirements: the On-The-Fly Dereferencing Pattern, the Crawling Pattern and the Query Federation Pattern. These patterns have been broadly described in [13], and they are briefly introduced in the following paragraphs to identify the alternative ways of consuming the Linked Data Cloud.

On-The-Fly Dereferencing Pattern is also known as follow your nose approach. This pattern conceptualizes the web as a graph of documents which contains dereferenceable URIs. So, an application execute a query by accessing an RDF file by dereferencing the URL adress then follows the URI links by parsing the received file on-the-fly [12]. The problem with this pattern is the performance of the complex operations especially when it is needed to dereference thousands of URIs.

The Crawling Pattern follows the approach of web search engines like Google, Yahoo. In this approach, collection of data and usage of the collected (cached) data are two seperate taks. So the data collection task constantly crawl the web by dereferencing URLs, followign links and integrating te discovered data on the local site. The main advantage of the crawling pattern is its performance. Applications can use high volume of integrated data in much higher performance than other patterns. On the other hand, the main disadvantages of this pattern is that original data may change while you use it from replicated local cache (stale data problem). Also, integration of the discovered data is a very challenging task since different publishers may use different URLs to identify the same concept. In this case, one has to resolve this identity problem and also to consider the quality of the data (using the provenence knowledge) while integrating the all collected data.

Query Federation Pattern is based on dividing a (complex) query into subqueries and distributing sub-queries to relevant datasets which are selected using metadata about datasets. It raises on the findings of the database literature on distributed query processing. Query federation is composed of two main steps before performing a query. Firstly, query is divided into sub-queries and datasets relevant with sub-queries are selected using some metadata which reflects dataset content. Then, the query execution plan is constructed using statistics about datasets in the query optimization step. For the purpose of executing sub-queries on distributed data sources, query federation requires accessing datasets via SPARQL endpoints. SPARQL endpoints have become the standard approach to publish high volume of data since all relational database wrappers and RDF stores support data publishing through endpoints. Also, puslishing metadata about the data sources was added to the "Linked Data - Design Issues" note as a new principle in 2009 and a RDF scheme called VoID (Vocabulary of Interlinked Datasets) was proposed [2] to express metadata about the data sources and has become a defacto standart. So, there is a well established infrastructure to execute Query Federation Pattern and it is the one of the most researched topic

within the linked data community. The main problem is again performance of the complex queries especially when query needs to join data from large number of data sources. But, recent federated query engines like SPLENDID [10], WODQA [1] show reasonable performances even with complex queries.

When these three pattern are examined, it can be easily noticed that Query Federation an On-the-fly Dereferencing patterns query data from its original data sources, on the other hand Crawler pattern brings and integrates all crawled data in the application's local site. At this point, an important question that need to be answer is which pattern(s) are more suitable for semantic agent and the Linked Data Cloud interaction. Let's remember the Semantic Web scenario again to answer this question.

Lucy's semantic agent looks up several lists of providers, and checks for the ones in-plan for Mom's insurance within a 20-mile radius of her home and with a rating of excellent or very good on trusted rating services. In the world of linked data web, it can be easily assumed that there are two datasets publishing medical providers knowledge (types of services, location ext.) and rating knowledge as SPARQL endpoints. There will be links in these datasets and also links from medical provider data to the insurence dataset published somewhere else. So, the answer of the question is clear, semantic agents first able to discover right dataset(s) and query linked data in their original place by using Query Federation or On-the-fly Dereferencing patterns. Of course, semantic agents can take role in crawling process like handling identitiy resolution, quality handling and management of high volume of crawled data, but agent researchers first need to find effective ways to incorporate linked data querying patterns with the agent systems.

3 A Conceptual Architecture for Linking Agents with the Linked Data Web

As it is discussed in the preceding section, semantic agents should be able to discover the relevant dataset(s) according to the agent's running task requirement(s) and then able to query these dataset as the part of task execution. But querying is not enough in situations where agent need to react to the changing conditions in the Linked Data Cloud. For example, ratings of medical provider may change and agent need to monitor these changes to inform Lucy about the fall of her medical service rating numbers. Besides of rating numbers, many parameters of surrounding the Linked Data Cloud such as new medical provider enterence, insurence conditions or laws, conditions of provider services may change over time and semantic agent may need to monitor all of these parameter depending on its requirements.

The Linked Data Cloud changes constantly in two dimensions. In first dimension, cloud itsself expands by enterence of new data sources and creations of new links between the sources. Keeping track of these changes is a must for a semantic agent to be able to discover right knowledge sources to query depending of its task's requirements. In the second dimension, actual data may change

in the data sources and these changes may be critical for agent's internal tasks (such as Lucy's medical provider information) and need to be monitored by the agent. Consequently, semantic agent has to monitor and interpret the changes in the Linked Data Cloud structure and also changes in specific data sources which are dependent to its task(s) requirements. It is obvious that handling all of these linked data dynamism within the agents makes agent implementation very complex and it is agains the well known software engineering best practice known as seperation of concerns. Agent researchers encountered this problem before and defined the environment abstraction [17] to cope with it.

Environment is defined as the first class entity used during the design of multi-agent systems, and it includes all the resources and services that an agent needs. Thus, environment shields the agent developer from the complexity of the outside world. In the similar manner, it is necessery to shield semantic agent developer from the complexity of handling the Linked Data Cloud dynamism. So, it is time to think how linked data dynamism can be managed within the environment abstraction and what kind of services (interfaces) of environment should be provided to semantic agents to simplfy the usage of the Linked Data Cloud.

To define internal structure of the proposed architecture, the well known A&A (Agents and Artifact) [15] meta-model will be used. Thus, to form a basis for our discussion on the proposed conceptual architecture, an overview of the A&A meta-model is given first. The artifact concept lies at the core of the A&A metamodel. Artifacts are the building blocks of the environment and provide specific functionalities for agents. An artifact has a usage interface which defines the operations that an agent can excute on that artifact. There are two types of actions provided for an agent that can be performed on an artifact. The first one is the use action through which the agent can execute the operations in the usage interface of the artifact. The other one is the focus action through which an agent can start to observe specific properties of the artifact. These two actions can be used by semantic agents to query the Linked Data Cloud and to monitor the chances in specific data source(s). Additionally, events generated as a result of the operations triggered by other agents can also be observed by a specific agent. Finally, the artifacts can interact with each other through their link interfaces. A workspace is defined as a logical container of agents and artifacts. It organizes agents and artifacts from a topological perspective and defines the scope for interacting with the environment. In terms of semantic agents perspective, workspaces defines the scope of a specific domain that a group of agents aim to interact. For example in Lucy's scenario, domain represent the medical domain and workspace knows and manages the linked data view of (national) medical domain. Of course, not just Lucy's agent but many agents may interact with this workspace and other domains are represented by different workspaces.

The proposed conceptual architecture has two layers as shown in Figure 1: Agent-Linked Data Integration Services (A&LDIS) layer and Linked Data Access Services (LDAS) layer. A&LDIS layer appears in every workspace. On the other

hand, LDAS layer can be used by all workspaces in the environment or can be replicated in different workspaces if data volume or performance considerations requires.The scope of the workspace is defined by the VoID documents stored in the VoID store about the dataset(s) in a specific domain. If there is one VoID store in the environment, then each workspace should be able to access the VoID documents in its domain of interest.

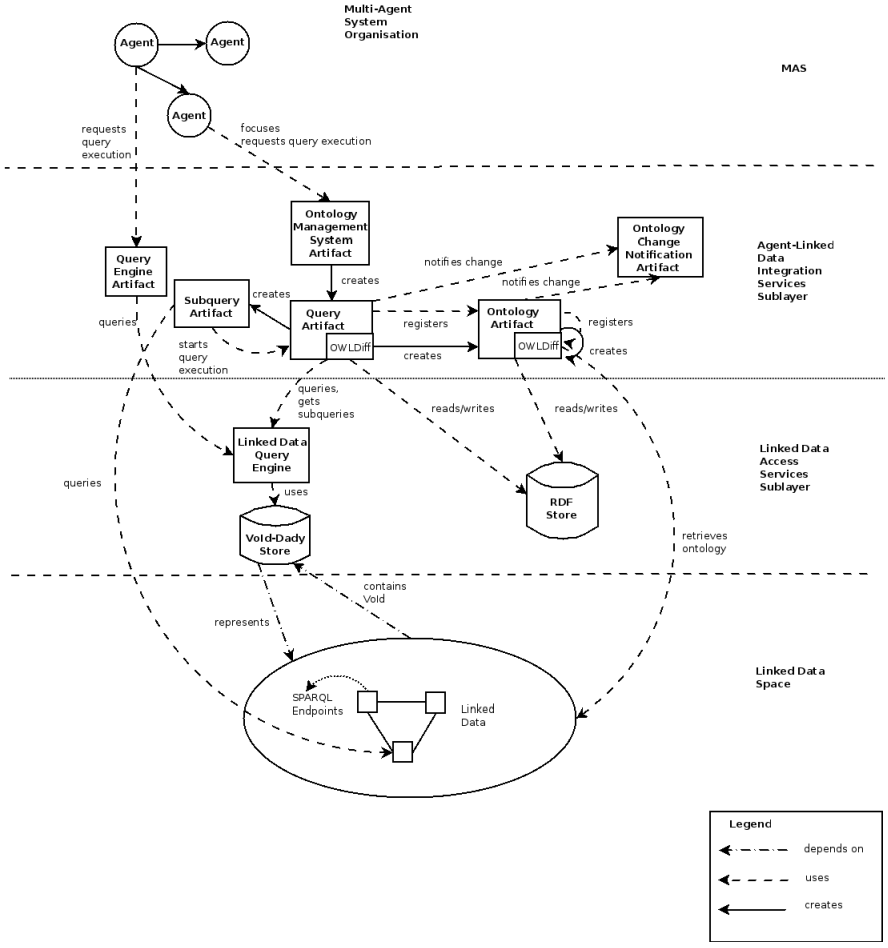

Fig. 1. Environment Architecture for Linking Semantic Agents with Linked Data Cloud

Semantics agents interact with the environment through two specially designed artifacts as seen in the figure. The fisrt problem is to define the basic entity(ies) that is passed by agents to artifacts, As it is discussed before, environment provides two basic services to agents: it executes queries and monitor changes of data that is requested by agent. For query service, it is obvious that

basic entity is SPARQL query. On the other hand, Monitoring service seems more complex since changes in one data source affect other sources if there are link(s) between them. But, SPARQL query can also be considered as the best entity for monitoring the Linked Data Cloud. Because, a SPARQL query defines a sub-graph (or a node as the minimal graph) of the Linked Data Cloud and any data (even the linked one) within the cloud can be defined with a SPARQL query. Changes in this sub-graph may occur because of any changes in linked datasets that constitute this sub-graph and observing changes in a SPARQL query lets semantic agents to be able to observe any kind of subgraph within the cloud. Therefore, semantic agents uses SPARQL queries for both querying and monitoring the Linked Data Cloud. It is important to emphasize that semantic agent only knows the ontology(ies) which are also in the environment's scope and construct SPARQL queries either for querying or monitoring based on its local knowledge. Note that accessing the URL address is not defined as one of the environment services, because it is a simple task for semantic agent to retrive and interpret an RDF file by itself.

Querying service is hadled by Query Engine Artifact. This artifact directly uses Linked Data Query Engine module located in LDAS layer to execute the query. Linked Data Query Engine module incorparates the Query Federation Pattern to the proposed archiecture. As discussed in section 2, Linked Data Query Engines execute SPARQL queries in two steps. In the fisrs step, query is divided to sub-queries and datasets relavant with each subquery are selected using the metadata about the datasets. In the proposed architecture, VoID vocabulary is selected to manage dataset's metadata since it is widly accepted as standart vocabulary to represent dataset metedata and there are well known Linked Data Query Engines such as SPLENDID [10], WODQA [1] which use VoID as the metadata vocubulary. In the second step, a query plan is constracted and sub-queries executed on SPARQL endpoints of selected dataset(s) and intermediate results are joint following the plan. So , semantic agent just deploys the query to the Query Engine Artifact, it then distributes it to relevent datasets according to its view of the cloud (based on its VoID store).

Proposed architecture includes only a Linked Data Query Engine for querying which means the exclusion of the On-The-Fly Dereferencing pattern from the architecture. This desicion depends on the fact that SPARQL endpoints become a defacto standart to publish high volume of data to the cloud. But, On-The-Fly Dereferencing algotihms found in the literature [11] can be incorparated to the architecture in a way that when there is no VoID description is found to execute the query, Query Engine Artifact can create an artifact that is responsible from On-The-Fly Dereferencing execution. This extansion makes the architecture more complex but more adaptable in terms of query execution.

Ontology Management System Artifact (OMSA) is responsible of monitoring service. To do so, it creates a seperate Query Artifact for each SPARQL monitoring request coming from registered semantic agent(s). Query Artifact first executes query with Linked Data Query Engine and stores the result in the RDF store of the environment. Then it divides the query to sub-queries and selects

the relevant dataset(s) using Linked Data Query Engine capabilities and create a seperate Sub-Query artifact for each identified sub-queries. Each Sub-Query artifact begins to monitor its query by periodically querying selected dataset(s) and compares the result with the previous one. Once a change is detected by a Sub-Query artifact, it is notified to the related Query Artifact. In this case, Query Artifact reexecute the SPARQL query using Linked Data Query Engine and compare the result with the stored result. If a change is detected, the result is notified to each registered agent(s) and store in the RDF store.

Monitoring the data changes of SPARQL query is not enough. Each SPARQL query depends on one or more ontologies and these ontologies may change dynamically (name of a concept or propert may change and/or new concept may added or removed ext.) in the Linked Data Cloud without the control of the semantic agent(s). Therefore, changes of ontologies, that monitored queries are build upon, should be monitored too. Query Artifact creates Ontology Artifact to monitor changes of ontologies. Ontology Artifact creates separate Ontology Artifact(s) for each ontology used in the query. Then, each Ontology Artifact monitors its assigned ontolog by preodically retrieving original ontology and identifies the changes (if any) using a ontology comparison algorithm like OWLDiff [14]. Ontological changes are notified to registered agents with Ontology Change Notification Artifact using a special ontology which is designed to represent changes within an ontology like the one proposed by Palma et. al. [16].

As a result, Lucy's semantic agent can query medical providers information and ratings from the Linked Data Cloud without any knowledge about the dataset(s) and how they are distributed and linked in the cloud. Moreover, agent may monitor any related information like changes in provider information, rating information, insurance policy or any changes in ontologies it depends on.

4 Conclusion

In this paper, an environment architecture is proposed to facilitate the interaction between semantic agents and the Linked Data Cloud. Proposed architecture is based on the experience of the implementation attempt of such an environment [9]. These efforts can be considered as initials steps to attract attention of both agent and linked data researchers to this challenging area of linking agents to the Linked Data Web.

References

1. Ziya Akar, Tayfun Gökmen Halaç, Erdem Eser Ekinci, and Oguz Dikenelli. Querying the web of interlinked datasets using void descriptions. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *5th Linked Data on the Web Workshop (LDOW 2012), 21th World Wide Web Conference (WWW 2012)*, Lyon, France, 2012.

2. Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing Linked Datasets - On the Design and Usage of voiD, the 'Vocabulary of Interlinked Datasets'. In *WWW 2009 Workshop: Linked Data on the Web (LDOW2009)*, Madrid, Spain, 2009.
3. Tim Berners-Lee. Linked data - design issues. http://www.w3.org/DesignIssues/LinkedData.html, 2006. [Online Technical Note; accessed March-2014].
4. Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web: Scientific American. *Scientific American*, May 2001.
5. Chris Bizer, Anja Jentzsch, and Richard Cyganiak. State of the LOD Cloud. http://www4.wiwiss.fu-berlin.de/lodcloud/state/, August 2011.
6. W3C Community. ConverterToRdf. http://www.w3.org/wiki/ConverterToRdf.
7. Richard Cyganiak, Christian Bizer, and Freie Universität Berlin. D2r server: A semantic web front-end to existing relational databases.
8. Souripriya Das, Seema Sundara, and Richard Cyganiak. R2rml: Rdb to rdf mapping language. http://www.w3.org/TR/r2rml/.
9. Riza Cenk Erdur, Oylum Alatli, Tayfun Gökmen Halaç, and Oguz Dikenelli. Monitoring the dynamism of the linked data space through environment abstraction. In *I-SEMANTICS*, pages 81–88, 2013.
10. Olaf Görlitz and Steffen Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In *Proceedings of the 2nd International Workshop on Consuming Linked Data*, Bonn, Germany, 2011.
11. Olaf Hartig. Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In *ESWC (1)*, pages 154–169, 2011.
12. Olaf Hartig, Christian Bizer, and Johann Christoph Freytag. Executing sparql queries over the web of linked data. In *International Semantic Web Conference*, pages 293–309, 2009.
13. Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, San Rafael, CA, 1 edition, 2011.
14. Petr Kremen, Marek Smid, and Zdenek Kouba. Owldiff: A practical tool for comparison and merge of owl ontologies. In *Proceedings of the 2011 22nd International Workshop on Database and Expert Systems Applications*, DEXA '11, pages 229–233, Washington, DC, USA, 2011. IEEE Computer Society.
15. Andrea Omicini, Alessandro Ricci, and Mirko Viroli. Artifacts in the a&a metamodel for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456, December 2008.
16. Raúl Palma, Peter Haase, Óscar Corcho, and Asunción Gómez-Pérez. Change representation for owl 2 ontologies. In *OWLED*, 2009.
17. Danny Weyns, Andrea Omicini, and James Odell. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5–30, February 2007.