Exploiting Aggregated Open Data from Smart Cities for the Future Internet Society

D3.3: SMART-FI Urban Ontology Report

Authors	Serdar Yumlu, Mehmet Kurt, Cihat Yigit, Ehsan Valizadeh
Institution lead	SAMPAS
Version	V1.0
Reviewers	Caner Tosunoglu (SAMPAS)
Work package	WP3
Task	T3.1
Due date	30/9/2017
Submission date	05/09/2018
Distribution level (CO, PU):	PUBLIC
Abstract	This report will cover several ontologies designed to describe the urban environment and its related data for SMART-FI and determine its strategy.
Keywords	Semantic Modeling, Smart Cities Ontologies

License information	This work is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/

Document Description

Document Revision History

Version	*Date*	*Modifications Introduced*	
		Description of change	*Modified by*
v0.1	15/03/18	First draft version and TOC	Serdar Yumlu (SAMPAS)
v0.2	20/03/18	Check the status and contributions	Mehmet Kurt (SAMPAS)
v0.3	28/03/18	Check the status and contributions	Cihat Yigit (SAMPAS)
V0.4	05/04/18	Check the status and contributions	Caner Tosunoglu (SAMPAS)
V0.5	10/04/18	Check the status and contributions	Ehsan Valizadeh (SAMPAS)
V0.6	11/04/18	Sent to reviewer	Caner Tosunoglu (SAMPAS)
V0.7	15/05/18	Check the status and contributions	Mehmet Kurt (SAMPAS)
V0.8	20/06/18	Check the status and contributions	Cihat Yigit (SAMPAS)
V0.9	15/07/18	Check the status and contributions	Caner Tosunoglu (SAMPAS)
V1.0	05/09/18	Overall revision of document and prepare final version to release	Ehsan Valizadeh (SAMPAS)

Table of Contents

Table of Figures

Table of Tables

Terms and abbreviations

RDF	Resource Description Framework
SPARQL	RDF query language
ETL	Extract, Transform and Load
LOD	Linked Open Data cloud
URI	Uniform Resource Identifier
OWL	Web Ontology Language
WP	Work Package

Executive Summary

In this deliverable we present methodologies to design an ontology based on SMART-FI data models as an objective of T3.1: "Designing several ontologies describe the urban environment and its related data".

This document also gives some details on tools related to exploiting Data using Linked Data applications. In particular, we give an overview of some of existing applications and introduce the main technologies that support implementation and development.

1 Introduction

Ontologies allow developers to reuse and share application domain knowledge using a common vocabulary across heterogeneous systems or environments. Therefore, ontologies do not only provide semantics and reasoning power to the data described in a given application but also increase the interoperability with other data. One good practice when developing ontologies is to reuse as much knowledge as possible since it increases interoperability by reducing heterogeneity across models and reduces development time.

1.1 About this deliverable

This deliverable gives an introduction to techniques and software stacks related to ontologies and linked data applications and also tools for "Mapping of Relational Databases to RDF". Starting with an introduction to general techniques about linked data, ontology development tools will be assessed and then we take a look at SMART-FI Ontology concept.

1.2 Document structure

Section 1 introduces the document; Section 2 discusses Smart Cities and related technical approaches; Section 3 describe Linked Data Applications; Section 4 covers Linked Data State of the Art; in section 5 we present tools for "Mapping of Relational Databases to RDF"; Section 6 explains "SMART-FI Ontology". Section 7 is conclusion of deliverable; finally, section 8 includes References.

2 Smart Cities Ontologies

There are many ontologies that can be used to describe cross-domain data or data in specific domains (e.g., smart cities, energy). Therefore, developers should reuse those models that have been already created, validated and deployed rather than spending resources reinventing the wheel

2.1 Ontology Engineering

The topics of ontology engineering are comprehensively described in (Guarino, 1998; Devedzic, 2002; Corcho, 2007), including methods and methodologies for the development of ontologies, ontology development process and lifecycle, ontology tools and languages. The importance of ontology engineering is recognized, but applications of ontology are still analysed more widely than ontology engineering itself. The attention is made to the following aspects: ontology learning, knowledge acquisition process, ontology merging, alignment, evaluation.

The concept Ontology still has different meanings among researchers and practitioners in Software engineering/Semantic Web areas. The conception partially depends on the fact, modelling or engineering of ontology is more accentuated. The methodologies for the manual development of ontology are underdeveloped. There are still much heuristics in the development process. The approach to using domain ontology in the development and delivery of educational resources enables support for this process, increasing adaptivity and interactivity on student-study material level. The development and research of adaptive learning environments is struggled with the high investment for development and maintenance problem. Besides other, the lack of interoperability in the field of knowledge management (knowledge discovery, knowledge representation, and reasoning) can be mentioned. The domain centric learning, in generally, and our approach, in particularly, are more suitable for formal studies, where subjects are unambiguously specified.

2.2 Approaches for Mapping of Relational Databases to R DF

2.2.1 Ontology learning from relational schemas

The overall framework of our ontology learning is presented in Figure 1. The input for the framework is data stored in relational database. The framework uses a database analyzer to extract schema information from database, such as the primary keys, foreign keys and dependencies, etc. Then the obtained information is transferred to ontology generator. The ontology generator will generate ontology based on the schema information and rules. At last, user can modify and refine the obtained ontology by the aid of ontology reasoner and ontology editor. The frame is a domain/application independent and can learn ontologies for general or specific domains from relational database.

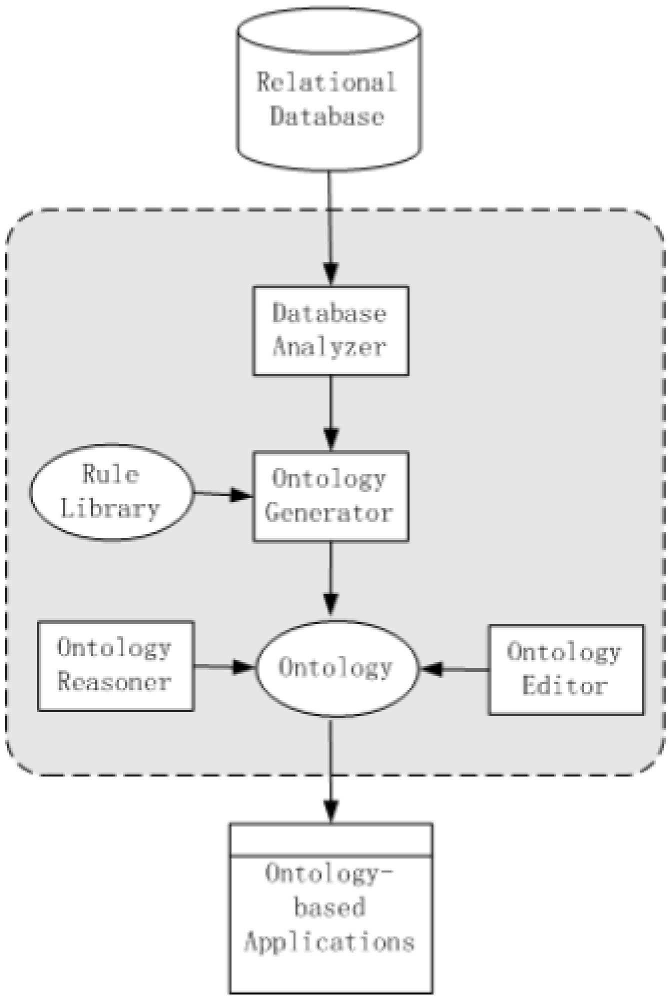

Figure 1: Ontology Learning Framework

2.3 Smart Cities Based on Web Semantic Technologies

Urban computing is a process of acquisition, integration, and analysis of a large amount of heterogeneous data generated by diverse sources in urban spaces, such as sensors, devices, vehicles, buildings, and humans, with the aim of addressing the major issues cities face (e.g., air pollution, increased energy consumption, and traffic congestion). There are three main challenges associated with urban computing: urban sensing and data acquisition, computing with heterogeneous data, and hybrid systems blending the physical and virtual worlds. In recent years, many applications have been proposed in different scenarios of smart cities including transportation, the environment, energy, social, the economy, and public safety and security. Moreover, the Intelligent Transportation Systems (ITS) have rapidly developed in Europe. Most of their work constructed features for specific domains and adopted machine-learning methods. In general, data obtained from smart cities are usually not such that they can easily be understood by humans. For example, the air quality recorded by sensors is usually represented by values such as “53” (PM2.5). These representations would be more meaningful if we had the semantic meaning of the numerical values such as “slight pollution”. Moreover, we can explain the results of some predictions by using common rules.

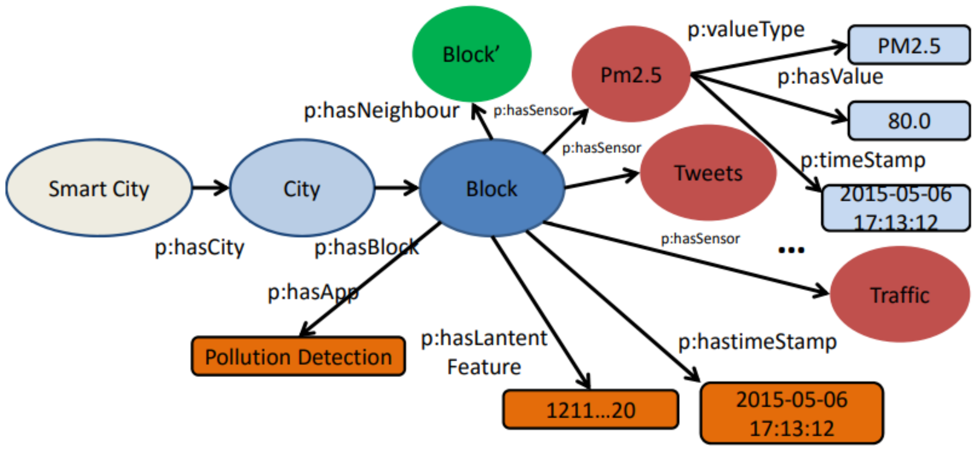

Figure 2 . Simple concept model of urban knowledge graph

3 Linked Data Applications

Based on a community-hosted collection of linked data applications and we will present a selection of linked data-driven Web application in the following. We have grouped them into four categories highlighting the main aspects, from a linked data usage point-of-view: (i) content reuse: applications that mainly reuse content of datasets in the LOD cloud in order to safe time and resources, (ii) semantic tagging and rating: applications that use HTTP URIs in the datasets for unambiguously talking about things, (iii) integrated question-answering systems: applications that focus on answering a user's question, and (iv) event data management systems: applications that allow people to organise and query event-related data. We note that this categorisation is a rough one. Many linked data-driven Web applications falling into several categories. However, it should help identifying the various use cases one can be after using linked data.

3.1 linked data-driven Web applications

3.2 Anatomy of a Linked Data-Driven Web Application

Following Bizer et. al., the approaches concerning linked data-driven Web applications can be categorised roughly into:

- Generic linked data browser56, such as Tabulator57 and OpenLink's ODE58, and Sigma59;
- Linked data search engines and indexer60 such as Falcons 61, and Sindice62;
- Domain-specific linked data applications.

As motivated above, we will focus on domain-specific linked data applications in this report. The reader is invited to refer to learn more about the first two categories. In the following, we will first give an overview of the components that can be identified throughout linked data-driven Web application. Then, we will discuss a concrete walk-through example and show how certain components may be realized. The section concludes in a brief overview of out-of-the-box components that help reducing development time.

3.2.1 Overview and Components

In a linked data-driven Web application conceptually one will—one way or the other—be able to identify the following components:

- A local RDF store, able to cache results and act as a permanent storage device to track users, etc. We note that an RDF store such as ARC2 or Virtuoso is not a strict requirement, though often it makes sense to manage the RDF data in a native environment.
- Some logic (a controller) and UI modules implementing the business logic, the User Interface (UI) and the interaction parts of the application. These components are not specific to linked data-driven Web applications, however typically required and found in the wild.
- A data integration component, focusing on fetching linked data from the Web of Data, either directly from the LOD cloud or via Semantic Indexer such as Sindice or Falcons.
- A republishing component that eventually exposes parts of the application's (interlinked) data on the Web of Data. It is a good practice to republish the application's data, hence providing again input to the LOD cloud.

The above listed components do not dramatically differ from what typical xAMP-based Web applications63 look like. The two main differences can be seen in the data integration and the republishing component. While the latter may well be a sub-module of the logic and/or UI module, the former characterized a linked data-driven Web application.

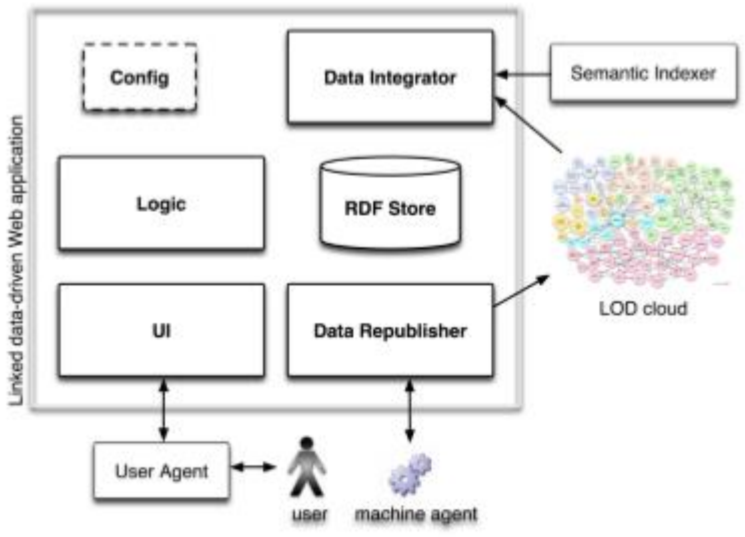

Figure 3 Concept of a linked data-driven Web application

In above figure the components and how they interact with external entities, such as indexer and user agents, are depicted

3.2.2 A Walkthrough Example

We will now use a hypothetical linked data-driven Web application, called "ssr" (pronounced like assessor) as a walkthrough example. In special, we will discuss two (in the context of this report) interesting components: the data integration component and the republishing component.

The ssr application's goal is, given a microblogging stream (such as available from the Twitter API64), to assess the expertise or interest of a certain user and make the so produced data available to humans and machines, respectively. The basic steps render as follow:

1. Retrieve a chunk of the stream via the site-specific API and RDFise it; this will typically result in RDF representations of the container of the actual message (e.g., represented using SIOC and FOAF). Additionally, extract tags from the posts (e.g., #linkeddata), called topics in the following;
2. Find datasets that are about the topics (via semantic indexer and voiD descriptions);
3. Query the datasets found in step 2. and retrieve details about the resources;
4. With the data from step 1. and step 2. create a new dataset that relates a person (and its social contact) with certain topics and expose it as linked data on the Web.

So far we have not shown how the components play together. Let us hence provide a system diagram and discuss the potential realization in greater detail as well as identify which modules in the ssr application match to which components of the concept of the linked data-driven Web application as introduced above.

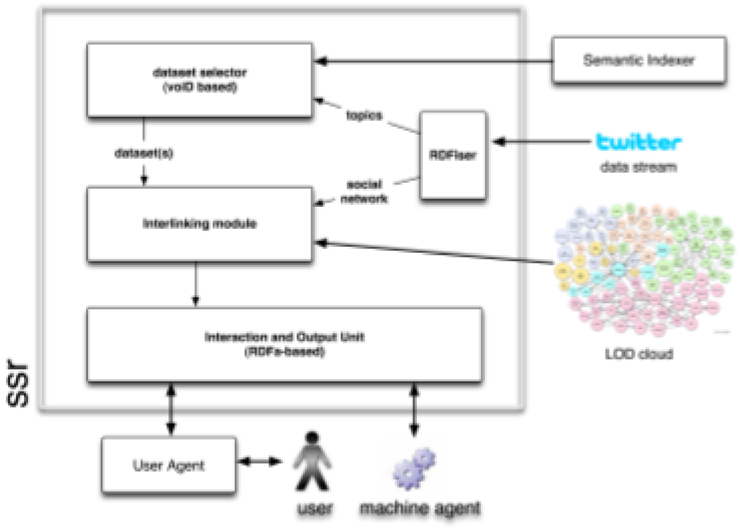

Figure 4 The system diagram of the ssr linked data-driven Web application

Above figure depicts the ssr system diagram; the modules' functions are according to the steps presented above.

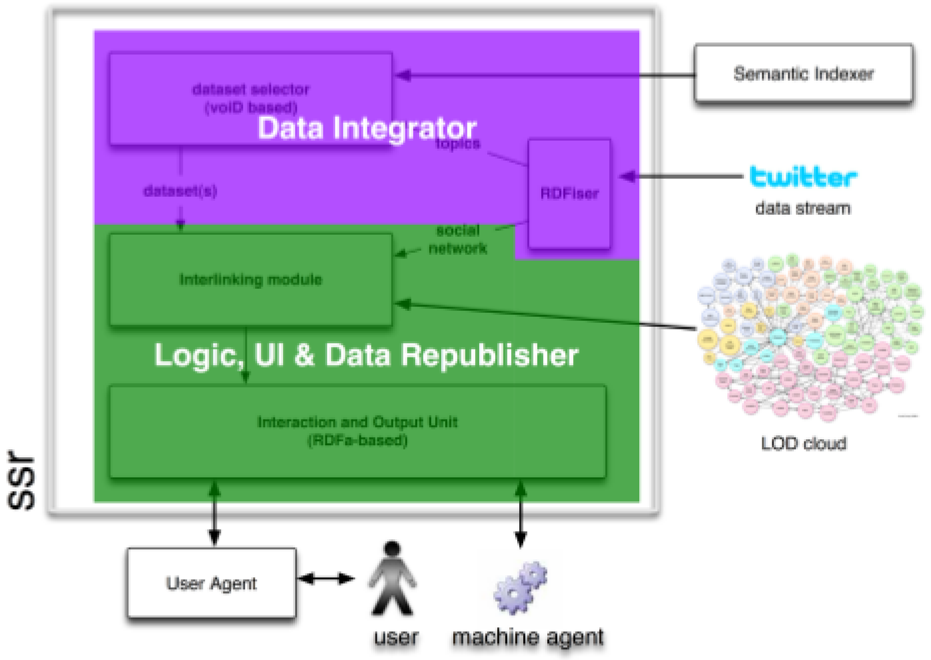

Figure 5 The system diagram of the ssr with the conceptual components overlaid.

Above figure depicts the ssr system diagram, overlaid with the conceptual components as described above. We note that no local RDF cache is used and that only the data integration component can be identified as a stand-alone functionality. The other modules map to the logic and UI equally as to the republishing component.

3.2.3 Out-Of-The-Box Components

We have now seen how a linked data-driven Web application can potentially look and work like. In the final section we want to point out some handy and time-saving tools and systems that can be used as a starting point or directly in the application development process:

- Systems that allow the consumption and publication of linked data out-of-the-box, such as Drupal with RDF CCK support or OpenLink's ODS;
- Linked data compliant cloud services, such as the Talis platform
- Systems that offer a high-level data integration API, such as Sigma
- Linked data validator and diagnostic tools, such as Vapour

3.2.4 Deployment and Access

While linked data in its early days suggested RDF/XML as the main deployment format, in the recent time RDFa has gained attention and is increasingly used. Our experience with RDFa-based linked data apps has shown that there is room for improvement regarding the RDFa deployment; the findings from this effort have been collected and will soon be available as a W3C note71.

4 Linked Data State of the Art

We will now have a more general discussion of the state-of-the-art concerning linked data by examinig good practices regarding the publication and the consumption of linked data.

4.1 Good Practices in Publishing Linked Data

We highlight good practices regarding linked data publishing in the following. As publishing is not the core topic of this report, we will only very roughly deal with it.

A first step in publishing is to design the URI space of the dataset. A more jargon term for this activity is minting URIs. The W3C note \Cool URIs Semantic Web and the linked data publishing guide gives detailed instructions and suggestions how to do this. We are confident that the above mentioned good practices are typically sufficient and just add that in case the data is exposed in RDFa, there are certain simplifications on the one hand, and some caveats on the other hand one should be aware of.

One of the central advises regarding vocabularies in the linked data world is: look at existing (widely deployed) vocabularies first and only invent new terms if you do not find appropriate terms there. However, this raises some questions: (i) what are widely deployed vocabularies, and where do I find them? and (ii) even if I think I have found a seemingly matching vocabulary? how do I know if a term is appropriate?

Regarding the first question, there are at least two known promising approaches: the publishing guide already covers some widely deployed vocabularies, such as FOAF or

SIOC and points to other, related resources. Secondly, one can use semantic indexer or dedicated service, such as the Talis' Schemacache or OntoSelect by DFKI to look up vocabularies. A sensible answer to the second question is likely to create a testbed, that is, to use and evaluate the vocabulary with concrete instance data. Though a rough check of, for example, the domain and the range of a property, or other ontological constraints is advisable as a first step, the ultimate "litmus test" is the usage of the vocabulary at hand. Once instance data is available, one can write, for example SPARQL queries and execute them, and/or perform reasoning with it. In most cases this reveals how usable the vocabulary actually is. Additionally, one should be aware of the fact that in most cases|especially true for the widely deployed vocabularies|there already exist widely used preifxes for the vocabulary namespace.

Another important aspect, which we will not further elaborate on in detail here is the RDFising and interlinking. Regarding the RDFising, that is, the process of creating an RDF representation out of a structured (XML, etc.) or non-structured (PDF) format using vocabularies as described above, we note that several tools and libraries exist36. Further, if one has a relational database available, one can use various RDB2RDF converter/mapper. For an overview on RDB2RDF mapper, see the respective survey of the W3C Incubator Group on this topic37. We note that there are out-of-the-box systems available that are able to create typed links based on a configuration. One example of such a system is Silk, an interlinking framework38. Finally, the interlinking can not only occur automatically, but also based on manual input by end-users (user-contributed interlinking).

4.2 Good Practices in Consuming Linked Data

Before we discuss the aspects of consuming linked data in greater detail, we will explain what we mean by "consuming" linked data. At least three distinct phases can be identified regarding the consumption process (even if the terminology might not be homogenous with respect to this):

1. **Discovery.** In the very beginning, one has either a term (e.g., "Paris") or, equally, a (set) of URIs identifying certain resource(s) and needs to _nd out more about it. That is, one essentially needs to look-up the URI that identi_es the concept (if the dataset is already known or no URI is given), or one needs to _nd a dataset that has the desired data;

2. **Access.** Once the dataset is identi_ed, one needs to know how to access it for a given task. For example, an indexer might prefer a di_erent access method than an aggregator or a bot that is interested in just a single resource;

3. **Processing.** Because a linked dataset follows the linked data principles, each entity in it (i)has a URI, and (ii) this URI is dereferenceable, that is, an HTTP GET against the URI will yield (potentially after a redirect) a representation of or information about the resource. This might, however, not be the _nal step in using linked data; one might now want to apply a SPARQL query or perform (lightweight) reasoning.

We note that these three steps can potentially occur in combination. Additionally we note that this is an iterative process, at least once the agent operates in the linked data space.

5 Semantic Web Security

Tim Berners Lee has specified various layers for the semantic web. At the lowest level one has the protocols for communication including TCP/IP (Transmission Control Protocol/Internet protocol), HTTP (Hypertext Transfer Protocol) and SSL (Secure Socket Layer). The next level is the XML (eXtensible Markup Language) layer that also includes XML schemas. The next level is the RDF (Resource Description Framework) layer. Next come the Ontologies and Interoperability layer. Finally at the highest-level one has the Trust Management layer. Each of the layers is discussed below

TCP/IP, SSL and HTTP are the protocols for data transmission. They are built on top of more basic communication layers. With these protocols one can transmit the web pages over the Internet. At this level one does not deal with syntax or the semantics of the documents. Then comes the XML and XML Schemas layer. XML is the standard representation language for document exchange. For example, if a document is not marked-up, then each machine may display the document in its own way. This makes document exchange extremely difficult. XML is a markup language that follows certain rules and if all documents are marked up using XML then there is uniform representation and presentation of documents. This is one of the significant developments of the World Wide Web. Without some form of common representation of documents, it is impossible to have any sort of communication on the web. XML schemas essentially describe the structure of the XML documents. Both XML and XML schemas are the invention of Tim Berners Lee and his consortium called the World Wide Web Consortium (W3C), which was formed in the mid 1990s.

Now XML focuses only on the syntax of the document. A document could have different interpretations at different sites. This is a major issue for integrating information seamlessly across the web. In order to overcome this significant limitation, W3C started discussions on a language called RDF in the late 1990s. RDF essentially uses XML syntax but has support to express semantics. One needs to use RDF for integrating and exchanging information in a meaningful way on the web. While XML has received widespread acceptance, RDF is only now beginning to get acceptance. So while XML documents are exchanged over protocols such as TCP/IP, HTTP and SSL, RDF documents are built using XML.

Next layer is the Ontologies and Interoperability layer. Now RDF is only a specification language for expressing syntax and semantics. The question is what entities do we need to specify? How can the community accept common definitions? To solve this issue, various communities such as the medical community, financial community, defense community, and even entertainment community have come up with what are called ontologies. One could use ontologies to describe the various wines of the world or the different types of aircraft used by the United States Air Force. Ontologies can also be used to specify various diseases or financial entities. Once a community has developed ontologies, the community has to publish these ontologies on the web. The idea is that everyone interested in ontologies of a community use the ontologies defined by the community. Now, within a community there could be different factions and each faction could come up with its own ontologies. For example the American Medical Association could come up with its ontologies for diseases while the British Medical Association could come up with its own ontologies. This poses a challenge as the system and in this case the semantic web has to examine the ontologies and decide how to develop some common ontologies. While the goal is for the British and American communities to agree and come up with common ontologies, in the real-world differences do exist. The next question is what do ontologies do for the web. Now, using these ontologies different groups can communicate information. That is, ontologies facilitate information exchange and integration. Ontologies are used by web services so that the web can provide semantic web services to the humans. Ontologies may be specified using RDF syntax.

The final layer is logic, proof and trust. The idea here is how do you trust the information on the web? Obviously it depends on whom it comes from. How do you carry out trust negotiation? That is interested parties have to communicate with each other and determine how to trust each other and how to trust the information obtained on the web. Closely related to trust issues is security and will be discussed later on. Logic-based approaches and proof theories are being examined for enforcing trust on the semantic web.

5.1 Security Issues for the Semantic Web

5.1.1 Overview of Security issues

We first provide an overview of the security issues and then discuss some details on XML security, RDF security and secure information integration, which are components of the secure semantic web.

As stated earlier, logic, proof and trust are at the highest layers of the semantic web. That is, how can we trust the information that the web gives us? Closely related to trust is security. However security cannot be considered in isolation. That is, there is no one layer that should focus on security. Security cuts across all layers and this is a challenge.

For example, consider the lowest layer. One needs secure TCP/IP, secure sockets, and secure With the semantic web, and especially with data mining tools, one can make all kinds of inferences. That is the semantic web exacerbates the inference problem. Recently there has been some research on controlling unauthorized inferences on the semantic web. We need to continue with such research (see for example).

Security should not be an afterthought. We have often heard that one needs to insert security into the system right from the beginning. Similarly security cannot be an afterthought for the semantic web. However, we cannot also make the system inefficient if we must guarantee one hundred percent security at all times. What is needed is a flexible security policy. During some situations we may need one hundred percent security while during some other situations say thirty percent security (whatever that means) may be sufficient.

5.1.2 XML Security

Various research efforts have been reported on XML security (see for example). We briefly discuss some of the key points.

XML documents have graph structures. The main challenge is whether to give access to entire XML documents or parts of the documents. Bertino et al have developed authorization models for XML. They have focused on access control policies as well as dissemination policies. They also considered push and pull architectures. They specified the policies in XML. The policy specification contains information on which users can access which portions of the documents. Users may have privileges associated with them depending on their roles. Furthermore, privileges may cascade down the documents. Privileges include read, write, append, distribute, and browse. For example, if a user has access to the root of a document then should his access, say read, propagate to all the descendants or the immediate children? In algorithms for access control as well as computing views of the results are also presented. In addition, architectures for securing XML documents are also discussed.

In the authors go further and describe how XML documents may be published on the web. The idea is for owners to publish documents, subjects to request access to the documents and publishers to give the subjects the views of the documents they are authorized to see. The idea is for the publishers to be untrusted. Essentially the owner specifies the access control policies. The publisher will then enforce the access control policies. Encryption algorithms as well as Merkel hash are used to ensure that the publisher is untrusted. Bertino et al provide the authenticity and completeness of the results computed by the publisher and sent to the subject.

5.1.3 RDF Security

RDF is the foundations of the semantic web. While XML is limited in providing machine understandable documents, RDF handles this limitation. As a result, RDF provides better support for interoperability as well as searching and cataloging. It also describes contents of documents as well as relationships between various entities in the document. While XML provides syntax and notations, RDF supplements this by providing semantic information in a standardized way.

The basic RDF model has three types: they are resources, properties and statements. Resource is anything described by RDF expressions. It could be a web page or a collection of pages. Property is a specific attribute used to describe a resource. RDF statements are resources together with a named property plus the value of the property. Statement components are subject, predicate and object. So for example, if we have a sentence of the form "John is the creator of xxx", then xxx is the subject or resource, Property or predicate is "Creator" and object or literal is "John". There are RDF diagrams very much like say ER diagrams or object diagrams to represent statements.

There are various aspects specific to RDF syntax. Iis very important that the intended interpretation be used for RDF sentences. This is accomplished by RDF schemas. Schema is sort of a dictionary and has interpretations of various terms used in sentences. RDF and XML namespaces to resolve conflicts in semantics.

More advanced concepts in RDF include the container model and statements about statements. The container model has three types of container objects and they are Bag, Sequence, and Alternative. A bag is an unordered list of resources or literals. It is used to mean that a property has multiple values but the order is not important. A sequence is a list of ordered resources. Here the order is important. Alternative is a list of resources that represent alternatives for the value of a property. Various tutorials in RDF describe the syntax of containers in more detail.

tements about other statements. For example, with this facility one can make statements of the form "The statement A is false" where A is the statement "John is the creator of XXX". Again one can use object-like diagrams to represent containers and statements about statements (see [RDF])

Now to make the semantic web secure, we need to ensure that RDF documents are secure. With RDF we need to ensure that security is preserved at the semantic level. The issues include what the security implications of the concepts resource, properties and statements. That is, how is access control ensured? How can statements, properties and statements be protected? How can one provide access control at a finer grain of granularity? What are the security properties of the container model? How can bags, lists and alternatives be protected? Can we specify security policies in RDF? How can we resolve semantic inconsistencies for the policies? How can we express security constraints in RDF? What are the security implications of statements about statements? How can we protect RDF schemas? These are difficult questions and we need to start research to provide answers. XML security is just the beginning. Securing RDF is much more challenging.

6 SMART-FI Ontology

Ontology development is not an easy task. It requires skills and is still an art rather than technology. People need a sophisticated methodology to help them develop an ontology. Although ontology building methodologies are not matured enough, there are some methodologies available. After a brief overview of some typical methodologies followed by a summarizing comparison of them, a set of finer-grained guidelines are presented in this section.

RDF(S)

RDF(Resource Description Framework) is a framework for metadata description developed by W3C(WWW Consortium). It employs the triplet model , well-known in AI community, in which object is called resource representing a web page. A triplet itself can be an object and a value. Value can take a string or resource. Object and value are considered as a node and attribute as a link between nodes. Thus, an RDF model forms a semantic network. RDF has an XML-based syntax(called serialization) which makes it resembles a common XML-based mark up language. But, RDF is different from such a language in that it is a data representation model rather than a language and that the XML's data model is the nesting structure of information and the frame-like model with slots.

Metadata is data about data. In the context of internet information processing, data is whatever is accessed by URL. Any internet resource contains information which is considered as an instance of a certain class. Using XML, you can mark up any piece of the original information in-line in which case an XML tag corresponds to a class whose instance is the thing marked-up by the tag. However, what RDF does is different. It creates a new representation in which it contains meta information which usually does not appear in the original resource, that is, metadata about the original information(data). For example, let us take an article as an original data. At the top, it usually contains a character string, say, "Riichiro Mizoguchi". If I mark it up as <author>Riichiro Mizoguchi </author> in the text, then it becomes explicit that the string denotes the author of this article. On the other hand, in the RDF representation of the metadata of the article might include the date when it was published which is not described in the article. Assuming the article is put at http://www.ei.sanken.osaka-u.ac.jp/pub/WI2001-Miz.pdf, the RDF description would be:

```
<rdf:Description rdf:about="http://www.ei.sanken.osaka-u.ac.jp/pub/WI2001-Miz.pdf">

 <author>Riichiro Mizoguchi</author>

 <pub-date>2001-10-23</pub-date>

</rdf:Description>
```

Although RDF has been designed for metadata representation model, it can be used as a general-purpose knowledge representation, which might be apparent from the fact that it is a kind of semantic network model.

RDF schema is a language to define tags(vocabulary) RDF uses. The most typical and common metadata such as the creator(author) of a resource and the date of its creation are defined in DC(Dublin Core) in which 15 metadata elements are defined. RDF schema does not have to define them for use in RDF but can borrow those 15 metadata elements with the name space: dc: Name space is the functionality given by XML and is used for designating a local world in which a set of tags are valid to avoid conflict between other tag sets. Although, at a first glance, the correspondence between RDF and RDF schema and that between XML and XML schema looks equal or at least similar, it is not true. The major role of XML schema is to constrain the instance to which a tag is attached. On the other hand, the major roles of RDF schema include giving tags with definition and their taxonomy to RDF, though it also specifies constraints of the possible values of the triplet. While XML is usable without XML schema, RDF is useless without RDF schema.

RDF schema has its built-in classes and meta-classes by which users can define any class and relation. Rdfs:Resource and its two subclasses: rdf:sClass and rdfs:Property are the key meta-classes. Every ordinary class defined in RDF Schema is an instance of rdfs:Class. In the same way, every property and relation defined in RDF Schema is an instance of rdfs:Property. For example, rdf:subclass-of is an instance of rdfs:Property and is a built-in relation. This shows that attributes and relations are not distinguished in RDF Schema. Relations and attributes are defined globally, that is, independently of any class unlike frame-based languages in which an attributes is defined as a slot of each class. This comes from DL conventions.

OWL (DAML+OIL)

Web Ontology Language(OWL) is also a language developed by W3C[OWL]. OWL is designed to make it a common language for ontology representation and is based on DAML+OIL[DAML]. OWL is an extension of RDF Schema and also employs the triple model. Its design principle includes developing a standard language for ontology representation to enable semantic web, and hence extensibility, modifiability and interoperability are given the highest priority. At the same time, it tries to achieve a good trade-off between scalability and expressive power.

Functionality related to the constraints for instances of a class include: unionOf for a Boolean operation of instance sets, disjointWith for mutual exclusiveness of classes, oneOf for enumeration of all instances, etc. Other functionality for constraints for property value include rdfs:domain and rdfs:range for restricting domain and range of a relations/property, minCardinality for constraining the number of values, transitiveProperty, inverseOf and so on. In most of the places where a class name is written, a class expression in terms of the Boolean operations of classes can be written to augment class specialization capability. Functionality for interoperability in the distributed environment of semantic web include sameClassAs, differentFrom, etc. to make it easier to export/import classes. In spite of its rich functionality, OWL is less powerful than the first order predicate logic in logical expression used in axiom writing, since such functionality is to be covered by the rule layer which is the next higher layer than the ontology layer in the layered cake[SW].

In the same example above, for example, if one wants to add some constraints, he/she has to use OWL. The following OWL code shows a constraint stating ConferencePaper and JournalPapers are mutually exclusive.

```
<owl:Class rdf:ID="JournalPaper">

 <rdfs:subClassOf rdf:resource="#Paper"/>
 <owl:disjointWith rdf:resource="#ConferencePaper"/>

 </owl:Class>
```

A property defined globally can be specialized for a specific class shown in the following where a relation “author” is restricted to have more than or equal to two authors when it is applied to CoAuthoredPaper class.

```
<owl:Class rdf:ID="#CoAuthoredPaper">

          <rdfs:subClassOf rdf:resource="#Paper"/>

          <rdfs:subClassOf>

                    <owl:Restriction>

                              <owl:onProperty rdf:resource="#author"/>

                              <owl:minCardinarity>2</owl:minCardinarity>

                    </owl:Restriction>

          </rdfs:subClassOf>

 </owl:Class>
```

Summarizing the above languages from knowledge representation(KR) point of view, they are within the paradigm which KR community has developed thus far. The new aspect is that they employ XML syntax to cope with web information processing. RDF(S) is a kind of semantic network. OWL is the same as RDF(S) in its data model and in the top-level ontology. The class rdfs:Property is a symbol level concept rather than an ontological concept. Therefore, RDF(S) does not distinguish between relations, attributes and features in spite of that all the three are essentially different. OWL does not provide users with adequate modeling facility for representing an ontology, though it is very appropriate for ontology interchange and sharing. In fact, an ontology is something scaffolding conventional knowledge representation onto the real world, that is, the fundamental structure of the world of interest, which require a sophisticated ontology theory. Ontology representation languages are expected to reflect the results of such ontology theories.

SMART-FI ontology consist of implementing and integrating several data models including FIWARE Data Models and other data sources to work together as discussed in D3.1 and publishing unified data models through apache Jena Fuseki to be accessible through HTTP.

7 Conclusion

This survey to document the current state of the art in mapping approaches between RDB and RDF was conducted to partially fulfill the objectives of the W3C RDB2RDF incubator group. To enable a coherent and effective comparison of the different RDB2RDF mapping approaches we defined a reference architecture for the survey consisting of six components such as mapping generation, query execution and data integration achieved by mapping RDB to RDF.

One of the important aspects of mapping RDB to RDF is the potential to explicitly model information that was either implicitly modeled or not represented at all in RDB, such as domain semantics. Hence, many of the tools and generic application in the survey have noted the importance of using a domain ontology, in addition to information from the RDB schema, in generating RDF. Another important aspect of mapping RDB to RDF is the potential for data integration by representing data from multiple RDB sources as a single RDF graph. The representation of mappings between RDB and RDF in a standardized form is necessary to enable their reuse and the RDB2RDF incubator group, in its final report [RDB2RDF XG Final Report] has proposed the use of the W3C Working Group Rule Interchange Format (RIF) to represent mappings. None of the projects reviewed in the survey use RIF to represent mappings between RDB and RDF.

Finally, this survey is expected to not only serve as a resource to the RDB2RDF incubator group but also for the community of researchers involved in mapping RDB to RDF in order to support the evolution of the Web of documents into a “Web of Data”.

8 References

SMART-FI Proposal 2016.

SMART-FI Project Consortium Agreement

EU Internet Handbook – Keep it simple

http://ec.europa.eu/ipg/content/tips/volume_en.htm

http://www.smart-fi.eu/

[1] eWork and eBusiness in Architecture, Engineering and Construction; Ardeshir Mahdavi, Bob Martens, Raimar Scherer - 2014

[2] LEARNING ONTOLOGY FROM RELATIONAL DATABASE; MAN LI, XIAO-YONG DU, SHAN WANG

[3] Semantic Framework of Internet of Things for Smart Cities; Ningyu Zhang, Huajun Chen,* Xi Chen, and Jiaoyan Chen

[4] Linked Data Applications; Michael Hausenblas, DERI Technical Report July 2009

[5] Exploiting Linked Data to Build Web Applications; Michael Hausenblas , Digital Enterprise Research Institute (DERI)

[6] A Survey of Current Approaches for Mapping of Relational Databases to RDF, W3C RDB2RDF Incubator Group January 08 2009

[7] Ontology development, tools and languages, Riichiro Mizoguchi

[8] Handbook on Ontologies, Steffen Staab, Rudi Studer

[9] Database and Applications Security: Integrating Information Security, Bhavani Thuraisingham

[10] Security Issues for the Semantic Web, Dr. Bhavani Thuraisingham

[11] Developing and Securing the Cloud, Dr. Bhavani Thuraisingham