INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Report: Semantic Web

MS Innovative Smart Systems
Academic year 2016-2017
GAUTIER, RENAUD
LOUBET, GAËL
PERSAND, KAVEENA
Supervisor :
SEYDOUX, NICOLAS

Contents

Introduction

As part of the Innovative Smart System (ISS) cross-curricular course, a training unit tackles analysis and processing of data in business applications.

The main goals of these practical sessions were to understand the key aspects of semantic web and ontologies, use ontologies, and understand the reasoner's deductions at different implementation phases of an ontology. To substantiate the concept, an ontology directed at a smart meteorological application was designed. We used ontologies to describe a weather station, so as to enrich collected data as far as we could.

For the allocated work sessions, two topics were tackled. The first one dealt with defining an ontology for weather, with "ad-hoc" sensors. The design of this ontology was segregated into two parts: definition of a lightweight ontology and addition of its individuals, and definition of a heavyweight ontology and addition of its individuals. This ontology could have been enriched with the use of SSN (a generic ontology for sensors and observations). The ontology editor Protégé was used for the implementation, and has lead to the observation of deductions of new knowledge by a reasoner, Hermit, based on the knowledge fed to it. This preliminary phase gave us some insight on the basic notions of ontologies and their components. The second part was dedicated to annotate 3-star open data into higher level open data. The open data used were meteorological observations made in Aarhus, Denmark. The objective set was to convert the data retrieved into 5-star data using the ontology defined in the first phase.

Despite this report being co-written, each one of us proceeded with our own implementation. The approach often differed and only the most established one will be presented in this document.

This report aims to convince of the adequate comprehension of the basic concepts and notions associated to semantic data processing by each member of the group. Hence, implementation choices and justifications will be favoured to implementation details.

1 Designing an ontology and using Protégé's reasoner

1.1 Designing an ontology

The first phase's goal was to create an ontology that would describe a meteorological event. The ontology should be able to make a connection between a phenomenon (rain, typhoon, ...) and its parameters (temperature, humidity, ...), but also between the measured parameter and the sensor that measured it.

1.1.1 Draft of the lightweight ontology

In order to express the different relationships, we used three kinds of concepts:

- Class: used to express a particular concept (phenomenon, place, weather, time, etc). Each concept can have its own subclass (e.g. "good weather" and "bad weather" as subclass of "weather"). The subclass inherits their parent's characteristics.

- Object Property: used to express the relationship between several classes. For example, the object property "measures" can be used to link an "Observation" to a "Parameter", so that an "Observation" "measures" a "Parameter".

- Data Property: used to express the type of data linked to a Class. For example, a data property "has a duration of" will link a Phenomenon to a integer value, such as a "Phenomenon" "has a duration of 35". We can also add a property on a data property (as it is also an OWL object), for e.g. "has a duration of" "is in minutes", in order to have more information about the relationship.

These tools were used to represent concepts such as "Place" (with subclass "Continent", "Country" and "Town"), "Observation", "Parameter", "Time" and "Value". Each having their own object and data properties.

Each entity has a unique ID (which was set to be automatically generated by Protégé). The defined entities were tagged with labels to be easily understood by humans. An entity may be associated with different labels, e.g. to take into account synonyms, and multilingual vocabularies.

1.1.2 Populating the lightweight ontology and the reasoner's deductions

Once the vocabulary has been defined, individuals may be added to the ontology. These individuals will be of the classes defined, and will have the object and data properties associated to them. For example, the individual "Toulouse" can be created as a "Town" and the object property "'Toulouse' 'is located in' 'France'" can be added.

After adding individuals to the ontology the reasoner can be run. Based on the set of rules defined, the reasoner can deduce new knowledge such as the class of an individual. It can also show the different steps undertook for its deductions and signal logical flaws should it find paradoxes or any incoherence with the defined axioms.

When the reasoner is run, new knowledge was inferred. As the object property "'Place' 'is located in' 'a Place'", has been applied on "Toulouse" and "France" respectively, the reasoner can deduce that "Toulouse" is a "Place" and "France" is a "Place". This produces new knowledge for the knowledge base; "Toulouse" and "France" belong to the class "Place".

1.1.3 Draft of the heavyweight ontology

To express more complex relationships, advanced options need to be used when creating the entities. For example, the "Disjoint" property is used to state that two classes are mutually exclusive ("Town", "Country", and "Continent" are "Places", but they are disjoint, i.e. an individual belongs to at most one of the mentioned classes).

In the same way, object properties offer several options that can be used to describe relations between objects. We used the "inverse of" option on the "characterises" object property. This property was used to link a "Parameter" with a "Phenomenon", but we wanted to express the case where "a 'Phenomenon' 'is characterised by' a 'Parameter'". Thus, the inverse option was used to link "characterise" with "is characterised by". Hence, if the data property "characterises" is set to link "Parameter" to "Phenomenon", its inverse property "is characterised by" is deduced by the reasoner to link "Phenomenon" to "Parameter".

Mathematical properties (namely transitivity, symmetry, reflexivity, etc) as may be added on object properties. For instance, we used the transitivity property on the "is located in" object property. Consider the case where a "Town" "is located in" a "Country", and that that "Country" "is located in" a "Continent", the reasoner should deduce that that "Town" "is located in" that "Continent". This was achieved by annotating the object property "is located in" as transitive.

Complex relationships can be defined using Manchester OWL Syntax. For example, to define a "Short Phenomenon" as being a "Phenomenon" that "has a duration of " less than or equal to "15 minutes", the following syntax was be used on the subclass "Short Phenomenon:

Phenomenon and ('has a duration of' some xsd:int[<= "15"^^xsd:int])

1.1.4 Populating the heavyweight ontology

When adding individuals to the heavyweight ontology, a few limitations of the vocabulary defined may come to light. For example, the individual "Monaco" is in real-life both a town and country, but if it is added as both "Town" and "Country" the reasoner signals an incoherence. The output of the reasoner is as seen in figure 1.

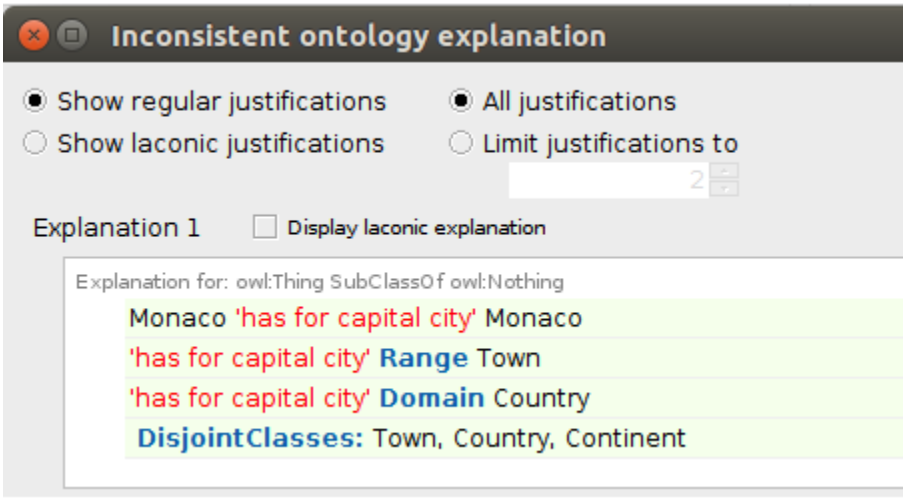

Figure 1: Incoherence in reasoner's deduction

1.2 Advanced design

This part describes integrate another ontology could have been integrated to extend our ontology. Due to time limitations, ontology extensions were not implemented. We wanted to integrate the concept of "Sensors" to our ontology. Re-using other ontologies instead of defining our own concepts each time is a step forward for better quality open data. Thus, SSN ontology, which represents the relationships between sensors, was chosen to be integrated. This ontology was going to enrich the description of the "Observation" class, while we would have extended their definition of "Station" by adding the notion of "Place".

Firstly, we would have had to add the labels defined in SSN ontology to our classes, so that "Observation" would have an "Observation value" label, that is, we would have had to manually align "Observation" from our ontology to "Observation value" from SSN. By doing this, the "Observation" class would have benefit from all the object and data properties related to "Observation value", including being linked to "Sensor Output" from SSN.

While the alignment could have been done manually, an alignment software could also have been used to automatically determine the corresponding classes and merge both ontologies. In case the labels were not enough meaningful for alignment, it would still have been possible to conduct alignment of both ontologies by taking into account semantic correspondences.

We could have easily integrated the "Device" class from SSN to our ontology by using its relationship with the class "Platform". In SSN, a "Device" "is attached to" a "Platform", so by adding the "is located in" object property to "Platform", the reasoner would have deduced the "Platform" as being a member of our "Place" class. This is how ontologies can be extended; by merging them while using labels and properties from both ontologies.

2 Enrich data using Java

Adding individuals one by one may be cumbersome, which is why part of the practical sessions was dedicated to adding individuals in an automated way. The ontology defined in the first part of the practicals was re-used. We did not use our own implementations for this part, but the one provided by the supervisor (which integrated SSN).

The data to be annotated was retrieved from the open database made available by the city of Aarhus. The retrieved data was in JSON format and could be regarded as 3-star open data. The data retrieved was to be enriched by adding meta-data. These meta-data included introducing the notion of "Observation" to the raw readings, adding the "Parameter" being observed, the "Sensor" producing the reading and the other semantic relations previously defined. The underlying idea was to exploit in Protégé the enriched dataset in order to verify the measures provided, and thus to discriminate erroneous values.

For example, to create an "Observation" that "measures" "Temperature", and that tht created "Observation" "has a value of" the extracted data, the following steps must be automated:

1. Extraction URIs of the entities "Observation" and "Temperature" from the ontology used

2. Extraction of the temperature reading from the open data file provided

3. An "Observation" individual must be created for each reading (e.g. "Obs0" for the first reading)

4. The URI of the object property "measures" must be extracted

5. The created "Observation" individual must be linked to "Temperature" through the object property "measures" (e.g. "Obs0" "measures" "Temperature")

6. The URI of the data property "has a value of" must be extracted

7. The created "Observation" individual must be linked to the extracted reading through the data property "has a value of" (e.g. "Obs0" "has a value of" "25.0" if the first temperature reading is 25.0)

These steps are obviously to be modified to accommodated all of the object and data properties to be linked to an Observation. The main idea for addition of object and data properties through the use of an application is exemplified in the above steps.

A Java application (using the provided template) was used to implement the extraction and rendering of data into a format compliant to our ontology.

To achieve the stated objective, two interfaces were implemented.

The first interface, IModelFunctions, was used to easily create individuals based on our ontology its add object or data properties to the created individuals. The implemented interface, DoItYourselfModel, used SemanticModel, which was provided with an interface to easily interact with our ontology. SemanticModel used JENA API, which is an API by Apache to interact with ontologies in Java.

The second interface, IControlFunctions, was used to easily create an "Observation" using the interface "IModelFunctions". The role of the Controller was to establish a link between the open dataset and the meta-data provided dictated the ontology.

A class, Controller, was implemented to use the implemented IControlFunctions interface, DoItYourselfControl, to:

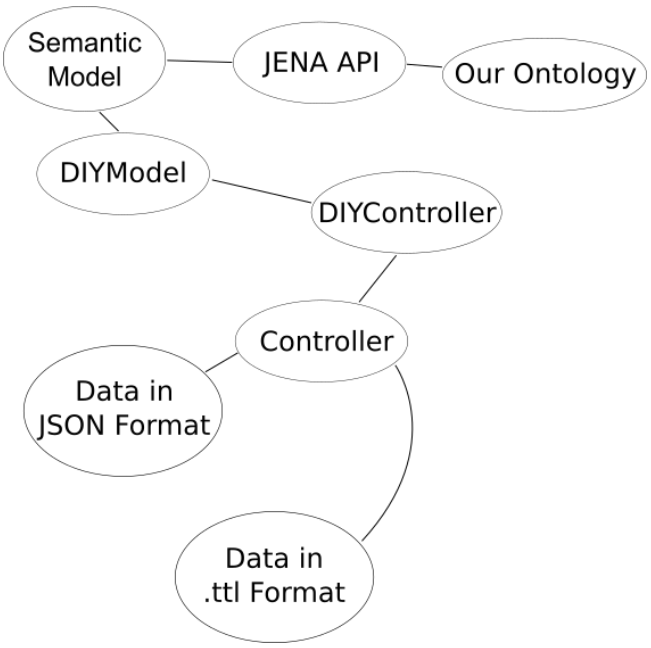

Figure 2: General interaction between the different objects used in the Java application that renders 3-start data into 5-star data

- Extract and parse data from the JSON file
- Instantiate an individual of type "Observation" for each reading, and add its properties (using the implemented interface, DoItYourselfControl)
- Export the semantic data created to a ".ttl" format

Amongst the object properties added to the created "Observation", was the object property linking an "Observation" to a "Sensor". Thus, the Controller manipulated data from the dataset by adding properties from the SSN ontology.

The transformed data was exported into a ".ttl" format. The exported file, contained the data retrieved and the information needed to understand the data.

The exported file was imported into Protégé to verify its adequacy.

We should have had checked the exported data to verify the efficiency and reliability of the sensors. According to the documentation provided by SSN, the classes "MeasuringCapability", "OperatingRestriction" and "EnergyRestriction" define the nominal operating conditions of sensors.

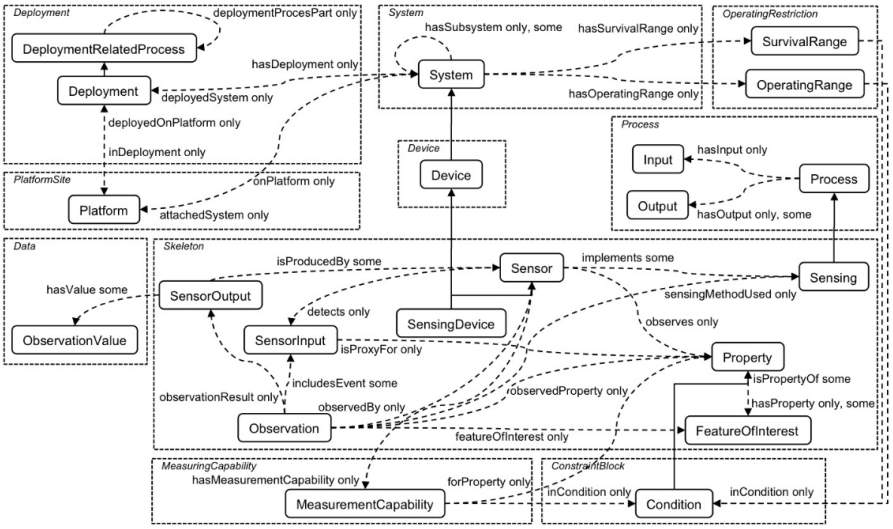

Figure 3: Overview of the Semantic Sensor Network ontology classes and properties

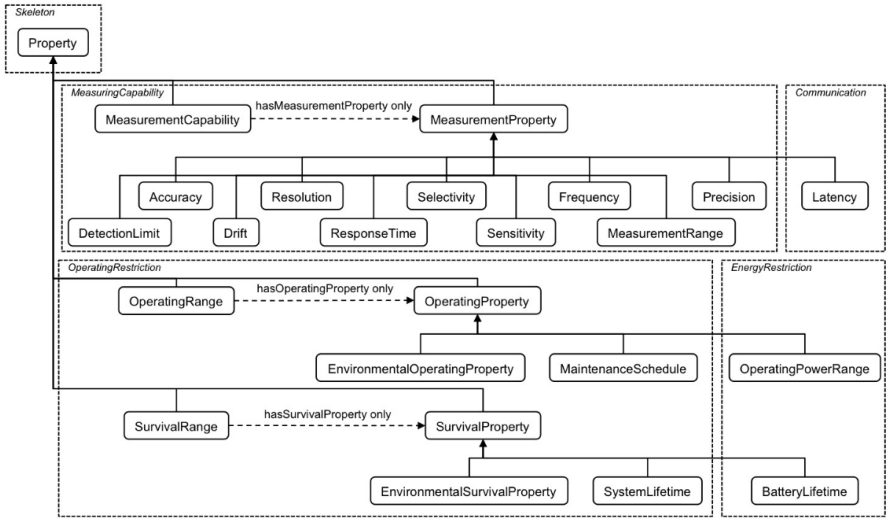

Figure 4: Enumeration of the measurement environmental and survival properties

Conclusion

Several aspects of semantic web were studied during these practical work sessions, through the use of ontologies with Protégé. We learnt how to create a new ontology, how to populate it and how to use it with a reasoner. The reasoner's output was used to deduce the nature of instances and correct the relations between classes. We also learnt how to integrate ontologies in a software, to enrich data and use the deduction power of the reasoner.

Through the work implemented during the second part of the practical work, we have seen how readings which would be difficult to be interpreted without any prior information was rendered into data easily understandable. The rendered data contained all the information needed to understand the readings and their associated context. These data may be easily imported by other entities and easily re-used. If the data are imported, SPARQL queries may be run on them to retrieve data specific to the user's need.

A Appendices

A.1 Classes and instances

Place	Continent	*Europe (instance)*	
	Country	*France (instance)*	
		Monaco (instance)	
	Town	*Toulouse (instance)*	
		Paris (instance)	
		Monaco (instance)	
Phenomenon	*A1 (instance)*	Bad Weather	*Fog (instance)*
			Rain (instance)
		Nice Weather	*Sunshine (instance)*
		Long Phenomenon	
		Short Phenomenon	
Parameter	*Temperature <=> Température (Instance)*		
	Atmospheric Pressure (Instance)		
	Rainfall (Instance)		
	Hygrometry (Instance)		
	Humidity (Instance)		
	Wind Power <=> Wind Speed (Instance)		
Observation	*P1 (Instance)*		
Instant	*I1 (Instance)*		

Table 1: List of classes and instances implemented in our ontology

A.2 Object Properties

Country	has for capital city	Town
Phenomenon	has symptom	Observation
Parameter	characterises	Phenomenon
Phenomenon	starts at	Instant
Phenomenon	is characterised by	Parameter
Place	is located in <=> has for location <=> is included in	Place
Observation	is made at <=> has for date	Instant
Phenomenon	ends at	Instant
Observation	measures	Parameter

Table 2: List of object properties implemented in our ontology

A.3 Data Properties

Observation	has a value of	(not specified)
Phenomenon	has a duration of	int
Instant	corresponds to timestamp <=> is characterised by	dateTimeStamp

Table 3: List of data properties implemented in our ontology

A.4 Implemented relations

I1	corresponds to timestamp	2015-11-10T10:00:00Z
P1	measures	Rainfall
P1	has a value of	3
P1	is located in	Toulouse
P1	has for date	I1
Toulouse	is located in	France
France	has for capital city	Paris
France	has for capital city	La Ville LumiÃĺre
France	is located in	Europe
A1	has symptom	P1

Table 4: Implemented relations in our ontology

A.5 Classes and instances, with SSN

Place	Continent	*Europe (instance)*		
	Country	*France (instance)*		
		Monaco (instance)		
	Town	*Toulouse (instance)*		
		Paris (instance)		
		Monaco (instance)		
	Platform	*Station1 (instance)*		
		Sonde_P1 (instance)		
		Sonde_T1 (instance)		
	Device	SensingDevice	*Sonde_P1 (instance)*	
Sensor			*Sonde_T1 (instance)*	
Phenomenon	*A1 (instance)*	Bad Weather	*Fog (instance)*	
			Rain (instance)	
		Nice Weather	*Sunshine (instance)*	
		Long Phenomenon		
		Short Phenomenon		
Entity	Quality	Property	Parameter	*Temperature <=> Température (Instance)*
				Atmospheric Pressure (Instance)
				Rainfall (Instance)
				Hygrometry (Instance)
				Humidity (Instance)
				Wind Power <=> Wind Speed (Instance)
Observation <=> Observation Value	*P1 (Instance)*			
Instant	*I1 (Instance)*			
Sensor Output	*Output_T1 (Instance)*			
	Output_P1 (Instance)			

Table 5: List of classes and instances implemented in our ontology after adding SSN

A.6 Object Properties, with SSN

Country	has for capital city	Town
Phenomenon	has symptom	Observation
Parameter	characterises	Phenomenon
Phenomenon	starts at	Instant
Phenomenon	is characterised by	Parameter
Place	is located in <=> has for location <=> is included in	Place
Observation	is made at <=> has for date	Instant
Phenomenon	ends at	Instant
Observation	measures	Parameter
Sensor Output	has for value	Observation
Device	is attached to	Platform
Platform	is located in	Place
Sensing Device	observes	Property
Sensor Output	is produced by	Sensor
Device1	is combined with	Device2
if Device1 is attached to Platform and Device2 is attached to Platform		

Table 6: List of object properties implemented in our ontology after adding SSN

A.7 Data Properties, with SSN

Observation	has a value of	(not specified)
Phenomenon	has a duration of	int
Instant	corresponds to timestamp <=> is characterised by	dateTimeStamp

Table 7: List of data properties implemented in our ontology after adding SSN

A.8 Implemented relations, with SSN

Station1	is located in	Toulouse
Sonde_P1	is attached to	Station1
Sonde_T1	is attached to of	Station1
Sonde_P1	observes	Rainfall
Sonde_T1	observes	Temperature
Output_P1	is produced by	Sonde_P1
Output_T1	is produced by	Sonde_T1
P1	is located in	Sonde_T1

Table 8: Implemented relations in our ontology after adding SSN

A.9 Deductions

A.9.1 After the implementation of the lightweight ontology

- "Toulouse" is a "Place"
  Explanation:
  "Toulouse" is a "Town" and "Town" is a subclass of "Place".

- "Paris" is a "Town"
  Explanation:
  "Paris" is a "Town" and "Town" is a subclass of "Place".

A.9.2 After the implementation of the heavyweight ontology

- "A1" is a "Rain"
  "A1" is a "Phenomenon"
  "A1" "has symptom" "P1"
  "P1" "measures" "Rainfall"
  "P1" "has a value of" "3"
  "P1" "takes place in" "Toulouse"
  "P1" "corresponds to timestamp" "I1"
  Explanation:
  "Rain" is an instance of "Bad Weather", thus "Phenomenon"
  "Rain" "has symptom" a "Rainfall" "Observation", with a positive value.

- "Paris" is equivalent to "La Ville Lumiere", "is located in" "France" and in "Europe". Moreover, "Paris" "is located in" "France"
  "Paris" is a "Town"
  "France" is a "Country"
  "France" "has for capital city" "Paris"
  "Europe" is a "Continent"
  "France" "is located in" "Europe"
  "France" "has for capital city" "La Ville Lumiere"
  Explanation:
  "is located in" is a transitive property.
  Subclasses of "Place" are mutually exclusive.
  "has for capital city" is surjective: there is exactly one capital city per country. A capital "Town" "is located in" the "Country", whose capital town it is.

- "Toulouse" is "Paris"
  If we add:
  "France" "has for capital city" "Toulouse", the reasoner deduces that "Toulouse" is equivalent to "Paris" as "France" can have only one capital "Town".

- "Monaco": a "Town" or a "Country"?
  Due to the "Disjoint" property on the subclasses of "Place", Monaco cannot be simultaneously defined as a "Town" and a "Country", contrary to reality.

A.9.3 After the completion of the ontology

- "Station1" is a "place" located in "France" and in "Europe"
  "Toulouse" is a "Town"
  "France" is a "Country"
  "Europe" is a "Continent"
  "Station1" is a "Platform"
  "Station1" "is located in" "Toulouse"
  "Toulouse" "is located in" "France"
  "France" "is located in" "Europe"
  Explanation:
  "is located in" is a transitive property
  Subclasses of "Place" are exclusive

- "Sonde_P1" "is located in" "Station1", in "Toulouse", in "France" and in "Europe", and "is combined with" "Sonde_T1"

- "Sonde_T1" "is located in" "Station1", in "Toulouse", in "France" and in "Europe", and "is combined with" "Sonde_P1"