LINKED ENERGY DATA, ENABLING MONITORING AND DECISION SUPPORT FOR IMPROVED ENERGY MANAGEMENT

Michel Böhms[1], Theo Rieswijk[2]

[1] TNO, Van Mourik Broekmanweg 6, Delft, The Netherlands, michel.bohms@tno.nl

[2] PRIVA, Zijlweg 3, De Lier, The Netherlands, theo.rieswijk@priva.nl

Keywords: Linked Data, Energy Management, Monitoring

Abstract

This paper shows how the European Odysseus project [1] reduces (non-renewable) energy/CO2 by improved decision making based on multi-level, semantic, energy configuration / monitoring data, fully based on open standards (W3C Linked Data approach/Semantic Web technology [2]) and an open source semantic server (Apache Marmotta [3]) implementing those open standards.

1 The European Odysseus Project

Odysseus stands for Open Dynamic System for Saving Energy in Urban Spaces, an ongoing European FP7 project in the scope of “Energy-positive Neighbourhoods”. The project coordinator TNO, the largest Dutch R&D institute, is also leading the energy modelling aspects complemented by the domain modelling by partner PRIVA, the leading Dutch energy solution provider for green houses and the built environment. CSTB, a French R&D institute is responsible for the software implementations, cGS, a Spanish IT consultant/developer and Advanticsys an expert in monitoring systems. Dissemination is managed by the Italian ESoCE Net and pilots are performed for the cities of Manchester and Rome. Main goal of the project is (“bad” or non-renewable) energy/CO2 reduction via awareness/monitoring and tactical/operational decision support that is economically feasible and not compromising peoples functions and comfort levels, in short: a sustainable solution (figure 1).

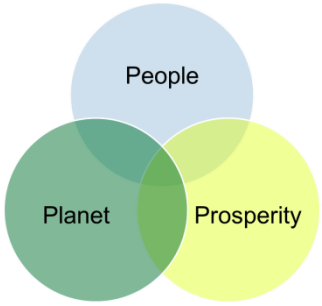

Figure 1 Sustainable solutions.

Reducing energy consumption can be typically accomplished via (in that order) (figure 2):

- Reduction of energy demand,
- Use of renewable energy resources, and
- Improvement of energy efficiency,

Odysseus focusses on the third bullet especially via improved decision making in the design and operational phases on Neighbourhood, Entity/Building and Device level. (figure 3).

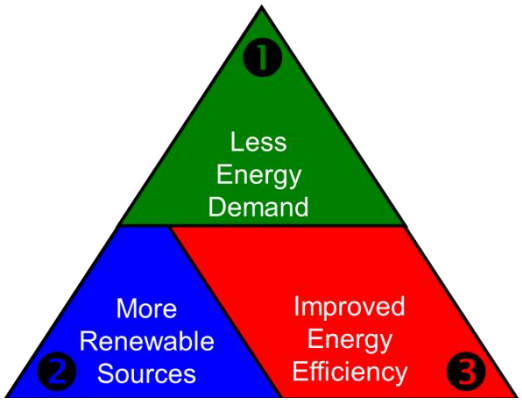

Figure 2 Measure types to reduce energy.

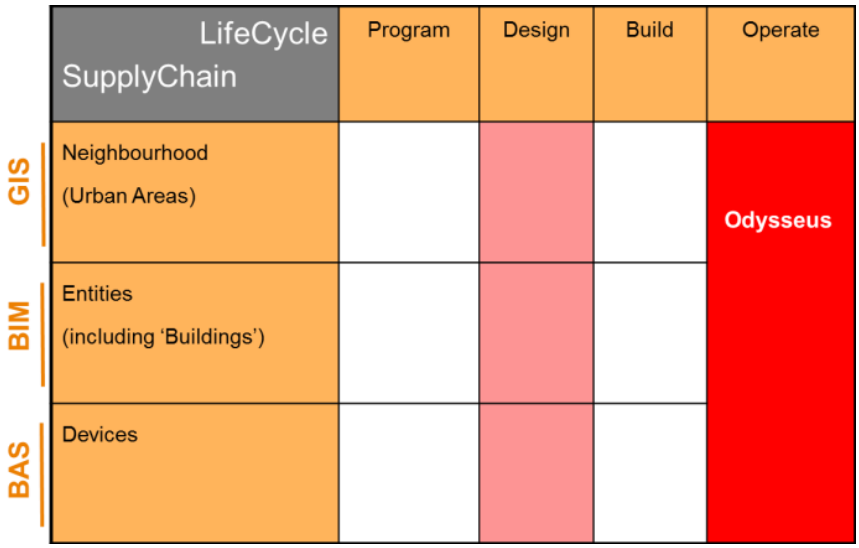

Figure 3 Life-Cycle and Supply-Chain scope.

The three levels addressed implicate we have to deal with a variety of existing ICT worlds including:

- GIS – Geo-spatial Information Systems
- BIM – Building Information Modelling, and
- BAS – Building Automation Systems, sensors, meters measuring conditions and (energy) flows

In this paper we focus on the open, semantic modelling of those three levels in an ontology referred to as dEPC – dynamic Energy Profile Card.

2 Modelling Approach

2.1 Introduction

This chapter describes a Modelling Guide (MG) for the development/definition of ontologies, sometimes also referred to as "object libraries". These ontologies are making use of a predefined "Concept Modelling Ontology"(CMO) generic upper ontology containing key modelling constructs for product modelling in general that are not yet part of the standard languages we use (OWL2, RDFS, RDF). CMO is a shared development by TNO (NL), CSTB (F) and RDF (BG).

2.2 Design & Implementation Principles

- Maximize reuse of existing open 'semantic web' standards (OWL2, RDFS, RDF, SPARQL, Turtle)
- Minimize own particular restrictions, extensions and modelling rules for maximum reuse
- Keep everything as simple as possible (but not too simple)
- Stick fully to the assumptions by the 'semantic web':
  - Open World Assumption (OWA): all OWL ontologies follow the Open World Assumption: all things not asserted are unknown (not false).
  - No (one) Globally Unique IDentifier (GUID) assumption. All identifiers (IDs) for OWL primitives (classes, properties, individuals) are never assumed to be Globally Unique for

those primitives. They can be unique names in themselves but one primitive might have more than one such identifier. OWL provides modelling primitives to state explicitly whether identifiers refer to the same or another resource (owl:equivalentClass, owl:equivalentProperty and owl:sameAs).

The set of extra modelling primitives is modelled as a generic 'upper ontology' called CMO distributed as a Turtle file: cmo.ttl. This ontology can be imported and reused by any other end-user ontology. The used prefix is "cmo". The CMO ontology itself is available at:

- http://www.modelservers.org/public/ontologies/cmo/cmo.ttl

The project specific ontologies (and data) using CMO are available at:

- http://www.modelservers.org/public/ontologies/odysseus (including HTML-variants for information).

All relevant ontologies and data sets are available for SPARQL querying at:

- http://marmot.tno.nl:8080/marmotta

2.3 Definitions

Ontology & Hierarchy

An ontology is an abstract, simplified view of a part of reality to be represented for some purpose. An ontology is essentially a set of Concepts, Properties and Relationships, or in OWL-terminology: a set of Classes, Datatype Properties and Object Properties respectively. Furthermore it contains Datatypes and all kinds of Restrictions (cardinality restrictions, universal and/or existential (owl:allValuesFrom / owl:someValuesFrom) logical restrictions, owl:hasValue restrictions, rdfs:domain and rdfs:range restrictions for properties etc.).

A hierarchy is a set of Classes or Properties connected by a specific Object Property that constitutes a partial or complete order between those classes. Such object property can be used to say that one class is 'higher/lower' than another class. Note: a hierarchy is not necessarily having a tree-structure: one higher class might be associated to 0, 1 or more 'lower' classes and one lower class might be associated to 0, 1, or more 'higher' classes. Often the interest goes to a hierarchy that constitutes a complete order: all classes of an ontology are part of the hierarchy, there are no "hanging classes", classes which have no higher/lower link with other classes.

Taxonomy & Meronomy

A taxonomy is a special kind of hierarchy where the object property connecting the classes is 'rdfs:subClassOf' with top-level class "owl:Thing" (the predefined 'most generic' class).

Another hierarchy example, not directly supported by OWL is a Meronomy where the object property connecting the classes is 'typicalPart'. In general the focus will be in the first place on taxonomies but meronomies are made possible via restrictions where all, in some way restricted properties are interpreted as "typical". In case of optional properties/parts, explicitly modelling minCard=0 (which implicitly is the default already) for a property, marks it as a qualified hasPart property. 'Typical' doesn't say anything about necessity! This approach is introduced because in OWL anything can be a part of anything else (when not constrained explicitly). Sometimes however, it is handy to make this (optional) relevance explicit so that the end-user can be given a template involving typical/relevant parts for an individual of certain class.

A taxonomy or meronomy or both typically form(s) the "backbone" of an ontology.

Decomposition

Explicit decomposition will only be applied on individual level by providing a predefined hasPart object property on class-level. As stated before, typical decomposition will be handled via qualified restrictions on this hasPart object property.

2.4 Main Modelling Guidelines

Language

The language chosen to define ontologies is: OWL 2.

Authoritative reference:

- OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation 11 December 2012, http://www.w3.org/TR/owl2-overview/.

Since we do not want to restrict ourselves w.r.t. expressivity we allow the “OWL 2 Full profile” to be used, knowing the consequences w.r.t. computational complexity and the limited support by reasoners.

OWL uses constructs from RDF Schema and RDF:

- Authoritative references: RDF Schema 1.1, W3C Recommendation 25 February 2014, http://www.w3.org/TR/rdf-schema/.
- RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014, http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/Overview.html.

Syntax

The syntax form (“format” or “serialization”) chosen in is: Turtle.

Authoritative reference:

- RDF 1.1 Turtle, Terse RDF Triple Language, W3C Recommendation 25 February 2014, http://www.w3.org/TR/turtle/.

Direct access

Beside import/export (aka upload/download) of Turtle files we assume direct access via the SPARQL query language.

Authoritative reference:

- SPARQL 1.1 Overview, W3C Recommendation 21 March 2013.

Natural correspondence (‘interpretation of OWL meta-concepts’)

- Concepts are modelled as OWL Classes
- Datatypes are modelled as OWL Datatypes (reusing underlying XSD Datatypes)
- Attributes are modelled as OWL Datatype Properties
  - In complex cases where “values” are really not sufficient one can always consider a complex property as a Concept and hence represent it as a Class (think material properties)
  - Some predefined attributes reflecting quantities: base quantities like length and time but also derived quantities like velocity, timesPerYear etc.
- Relationships are modelled as OWL Object Properties
  - One predefined relationship: hasPart
- Constraints are modelled as OWL Restrictions

Class/property naming conventions

- A class name always starts with a capital (upper case) letter (“CamelCase”)
- A property name always starts with a non-capital (lower case) letter (“camelCase”)
- Class and property names are in English.
- Classes and properties can have multi-lingual labels using rdfs:label
- In case of multiple labels in the same language skos:prefLabel can be used in case one wants to visualize the taxonomies in multiple languages , the preferred labels are then the ‘determined choice’ for the tree-node names per language.
- Character names according to RDF1.1 Turtle production rules (UTF-8)
- Class and property names are singular.

EXAMPLE

```
:Tunnel
  rdf:type owl:Class ;
```

```
        rdfs:subClassOf :CivilInfraArtefact .
```

We can also subclass while giving the exact discriminating property value (Dutch example):

```
:terrainType
  rdf:type     owl:DatatypeProperty ;
      rdfs:range  [ a rdfs:Datatype ;
          owl:oneOf  ( "ground"^^xsd:string "water"^^xsd:string )] .
:Bridge rdf:type owl:Class ;
      rdfs:subClassOf  :SpanStructure ;
      rdfs:subClassOf  [ a owl:Restriction ;
           owl:hasValue    "water"^^xsd:string ;
           owl:onProperty  :terrainType] .
```

It is also possible to use ‘range-like’ discriminators (not just specific values):

```
:CheapTunnel rdf:type owl:Class ;
 rdfs:subClassOf  :Tunnel ;
 rdfs:subClassOf  [ a owl:Restriction ;
          owl:allValuesFrom  [ a rdfs:Datatype ;
          owl:onDatatype xsd:float ;
            owl:withRestrictions  ( [ xsd:maxExclusive "6000000"^^xsd:float ] )] ;
            owl:onProperty     :cost] .
```

Individual decomposition

We predefine an instance decomposition object property:

```
:hasPart
 rdf:type owl:ObjectProperty .
```

This property is not transitive! It defines the direct parts of a whole. The transitive variant can be easily defined and derived/inferred if needed. We choose the direct variant for not “counting double” when traversing all parts of a whole in total cost, total energy etc. calculations.

EXAMPLE

```
:Deck_1
  rdf:type :Deck .
:Bridge_1
  rdf:type :Bridge ;
  cmo:hasPart :Deck_1 .
```

Typical decomposition

As stated earlier, typical decomposition is modelled at classes via restrictions on the hasPart object property in a qualified way. The interpretation is that any property restricted in some way (min/max cardinality, allValuesFrom, someValuesFrom, hasValue…) is by definition relevant for the class hence a “typical”

property. For optional properties without constraints we explicitly model explicitly “minCard=0” to indicate this fact.

EXAMPLE

```
:Bridge rdf:type owl:Class ;
  rdfs:subClassOf  [ a owl:Restriction ;
      owl:minQualifiedCardinality          "0"^^xsd:nonNegativeInteger ;
      owl:onClass :Deck ;
      owl:onProperty cmo:hasPart ] .
```

Typical properties

In OWL ontologies Classes and Properties are fully independent. In other modelling language one typically defines properties in the context of a class (like in EXPRESS: ‘Explicit Attributes’ in the context of an ‘Entity’). Not so in OWL: any property can be a property of any class, unless restricted explicitly (i.e. via a domain restriction for a property). This holds for both datatype properties and object properties. The same approach we used for typical parts we apply to typical properties (now non-qualified, i.e. no use of “owl:onClass”):

EXAMPLE

```
:Bridge
  rdf:type owl:Class ;
  rdfs:subClassOf  [ a owl:Restriction ;
       owl:minCardinality
         "0"^^xsd:nonNegativeInteger ;
       owl:onProperty :height ] .
```

This approach can be used for both datatype properties and object properties (standing for typical relationships in that case).

3 Energy Ontology

3.1 Introduction

The Odysseus dynamic Energy Profile Card (dEPC) ontology or simply the “Odysseus Energy Ontology”, is an advanced information structure covering a variety of information types relevant for improved energy-efficiency in a city’s neighbourhoods covering entities like buildings where people perform functions (executing processes, activities, actions, ...) enabled/supported by functionalities provided by appliances under the right indoor climate conditions:

- Energy: roles like producing, consuming, storing energy, played by: …
- Matter: physical things on two aggregation levels like:
  - Entities like buildings or windmills but also
  - Devices like sensors, measuring space conditions; energy meters, measuring energy flows, and actuators like a radiator that condition: …
- Space: conditional areas like building spaces (optionally grouped into zones, externally influenced by outdoor climate/weather).

These three aspects (energy, matter, space) are detailed in the next sections.

3.2 Energy

An energy network is a set of energy nodes and energy connections, where each energy connection connects two energy nodes with a direction. Energy nodes comes in five flavours:

- Energy production nodes,
- Energy consumption nodes
- Energy logistic nodes like switch nodes, controlling the flows between energy nodes (blocking certain connections and hereby stopping an energy flow or making the energy flow take a specific path in case of multiple choices)
    - Energy storer nodes where energy can stay over time, and
    - Energy transformer nodes, which do not consume or produce energy but just transform the energy for the same energy form: i.e. high voltage to low voltage.

The energy connections indicate how energy nodes are topologically connected and determine how energy can flow from one node to another.

Energy Networks (including their energy nodes and energy connections) have a specific energy form. We distinguish:

- Electricity
- Gas ("chemical")
- Heat ("thermal")
- Wind ("kinetic")
- Solar ("radiance")

3.3 Matter

Like energy nodes we have "material nodes" that have a relevance in energy management. Sometimes they fulfil one or more energy roles, sometimes they measure conditions (sensors) or energy flows (meters).

Material nodes can be seen on multiple levels of aggregation of which we select two:

- Neighbourhood Level, where the energy node roles are played by the (energy) installations of Entities like buildings, windmills, systems etc., and
- Entity Level, like the Building Level, where the energy nodes roles are on a lower detail level and played by Devices like appliances or installation parts like energy infusors (like radiators) or energy extractors (like generators)

The cities and neighbourhoods can also have energy consumption and production etc. as a whole but are not modelled explicitly (they are not considered playing an energy node role in some higher aggregation level). Energy properties for these levels can be obtained by querying the existing levels on-the-fly.

A building as special kind of entity has typically various installations covering various energy forms including electricity, gas and heat, that all consists of zero or more devices. The hierarchy of the decompositions of the energy networks having different energy forms is at building level strongly correlated with the energy node interaction of the energy flows in the energy connections. This makes it possible to state that the 'energy consumption-production' of a certain form by a system is always the summation of the 'energy consumption-production' of its direct parts. Examples of special kinds of infusor devices are:

- Radiators, increasing the temperature of a building space,
- Cooling Systems, decreasing the temperature of a building space,
- Lightings, increasing the illuminance of a building space,
- Blinds, decreasing the illuminance of a building space,
- Humidifiers increasing the relative humidity of a building space.

Sensors and Meters

Outdoor climate/weather conditions of the neighbourhood where the entity (like a building) is located, or indoor climate conditions of building spaces can be measured by sensors. The mentioned infusors/extractors effecting the indoor climate conditions of associated building spaces but also other devices like appliances and meters/sensors themselves consume and/or produce energy. This energy consumption/production can be measured indirectly by energy meters that are associated to energy flows as signals for energy connections between energy nodes.

A special case of Signal is a ProfileSignal being a discrete time series of TimePointSignals (being themselves Signals and related to simple 'base' or more complex 'derived' quantities (having units) with a time stamp) where quantities refer to the above mentioned space conditions and energy node/energy connection energy properties (energy consumption, production, flow etc.). For each signal we can optionally indicate:

- valueType: Required, Expected or Measured (default: Measured),
- valueKind: Instant, Incremental, Accumulative, Average , Minimal" or Maximal,
- valueOrigin: Asserted or Inferred (default: Asserted).

3.4 Space

Spaces are outdoor or indoor areas (referred to as building spaces) which typically need certain conditions for activities that have to be performed in them. This is why we refer to them in this context collectively as “conditional areas”. Buildings consist of one or more building spaces as conditional area. Furthermore, buildings can have zero or more (horizontal, vertical or mixed) zones that group a minimum of two building spaces into zones. Note that grouping is not the same as (strong) decomposition: when a zone is deleted, the rooms are still there; also the building spaces can be “part of” multiple zones.

A building space or outdoor area has a set of optional conditions that together form its indoor/outdoor climate like temperature, relative humidity, CO2 level and illuminance. These conditions are (given structural factors like wall types, roof isolation, room/zone topology, glazing types) determined by, external dynamic factors like the outdoor climate/weather conditions for the relevant neighbourhood, internal dynamic factors (occupation, appliances used) and finally the actuators/extractors. Next to the ‘entity-level’ conditional areas, we also distinguish ‘device-level’ conditional areas.

Energy Converters are special cases that transform one form of energy into another form of energy involving infusors, extractors and an actual carrier. Examples of convertors are:

- GasBoilers (gas > heat)
- Combined Heat Power (CHP) systems (gas > heat + electricity)
- SolarBoilers (solar > electricity)
- WindMill (kinetic > electricity)

All this knowledge is encode in a dEPC Ontology (or “Energy Ontology”) according to the modelling guidelines of section III using the TopBraid Composer ontology editor (Figure 4).

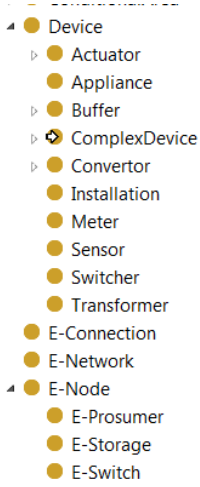

Figure 4 Taxonomy fragment of the dEPC ontology

4 Semantic Server (OSS APACHE MARMOTTA)

The dEPC ontology is uploaded to a Semantic Server. This server acts as a SPARQL-endpoint which can answer queries over the web. The chosen open source server is Apache Marmotta although alternative experiments are performed using Stardog and Joseki/Fuseki. Beside the ontology we filled the triple store with example data for both a neighbourhood-level and a building-level situation.

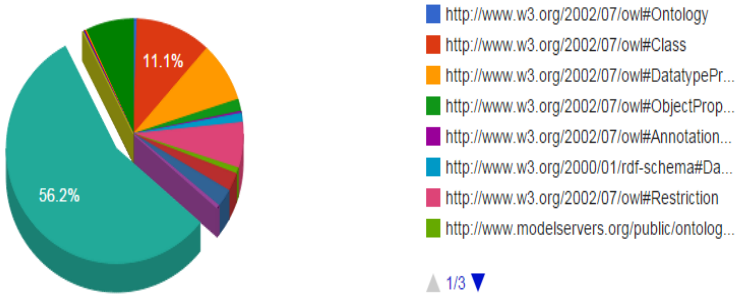

Figure 5 Marmotta data overview.

Hereby a small fragment of the Turtle file for the neighbourhood:

```
:Neighbourhood_1
  rdf:type depc:Neighbourhood ;
  cmo:hasPart :Battery_1 ;
  cmo:hasPart :Building_1 ;
  cmo:hasPart :Building_2 ;
  cmo:hasPart :Building_3 ;
  cmo:hasPart :Building_4 ;
  cmo:hasPart :Building_5 ;
  depc:playsRoleOf :Prosumer_1PE ;
  depc:playsRoleOf :Prosumer_1S ;
```

In the details we find the actual energy consumption data encoded as ProfileSignalss containing data for time points according to the specified interval:

```
:TimePointSignal_B1-P22
  rdf:type depc:TimePointSignal ;
  depc:eProduction "0.00"^^xsd:float ;
  depc:timeStamp "2014-05-18T22:00:00Z"^^xsd:dateTime .
:TimePointSignal_B1-P23
  rdf:type depc:TimePointSignal ;
  depc:eProduction "0.00"^^xsd:float ;
  depc:timeStamp "2014-05-18T23:00:00Z"^^xsd:dateTime .
```

We can query all this data via the web-interface of Marmotta where all data can be visualized (figure 6) and queried (figure 7) at **Errore. L'origine riferimento non è stata trovata.**.

Figure 6 A simples SPARQL query.

```
{
  "head": {
    "vars": [
      "subject"
    ]
  },
  "results": {
    "bindings": [
      {
        "subject": {
          "type": "uri",
          "value": "http://www.modelservers.org/public/ontologies/odysseus/building-examp
le.ttl#Room_1"
        }
      },
      {
        "subject": {
          "type": "uri",
          "value": "http://www.modelservers.org/public/ontologies/odysseus/building-examp
le.ttl#Room_2"
        }
      },
      {
        "subject": {
          "type": "uri",
          "value": "http://www.modelservers.org/public/ontologies/odysseus/building-examp
le.ttl#Room_3"
        }}]}}
```


Figure 7 SPARQL results (here in in JSON-LD-format).

In figure 7 the results from the SPARQL query are depicted in JSON-LD format showing the three rooms available in the data. These room can be used as context for further energy-related questions slicing and/or aggregating the energy consumption/production time series available.

5 Conclusions

The Linked Data approach involving Semantic Web technologies seems to be the ideal (open, flexible, well-defined, powerful) approach for modelling energy-related aspects of neighbourhoods and its relevant entities like buildings. The resulting ontologies and data are fully independent of any software application or platform and can be utilized in many ways including monitoring and decision support for energy management resulting in reduction of energy consumption and CO2 emissions on multiple scale levels.

The actual application scenarios involving Key Performance Indicators (KPIs) and Energy Conservation Measures (ECMs) are described in a separate Odysseus paper.

6 Abbreviations Used

BAS	Building Automation Systems
BIM	Building Information Modelling
CMO	Concept Modelling Ontology
dEPC	dynamic Energy Profile Card
ECM	Energy Conservation Measure
GIS	Geo-spatial Information Systems
JSON-LD	JavaScript Object Notation for Linked Data
KPI	Key Performance Indicator
LD	Linked Data [W3C]

MG	Modelling Guide
OSS	Open Source Software
OWL	Web Ontology Language [W3C]
RDF	Resource Description Framework [W3C]
SPARQL	SPARQL Protocol and RDF Query Language
Turtle	Terse RDF Triple Language [W3C]
W3C	World Wide Web Consortium

7 References

[1] http://www.odysseus-project.eu/

[2] http://www.w3.org/standards/semanticweb/data

[3] http://marmotta.tno.nl:8080/marmotta/

[4] http://odysseus.tno.nl:8080/webprotege/