Project ICT 287534
Start: 2011-09-01
Duration: 36 months
Co-funded by the European Commission within the 7th Framework Programme

SEMANCO Semantic Tools for Carbon Reduction in Urban Planning

SEMANCO

Deliverable 4.4
Interfaces with external tools

Revision: 8
Due date: 2013-08-31 (m24)
Submission date: 2013-10-25
Lead contractor: FUNITEC

Dissemination level		
PU	Public	X
PP	Restricted to other program participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Deliverable Administration & Summary					
No & name	**D4.4 Interfaces with external tools**				
Status	Final	**Due**	m24	**Date**	2013-10-25
Author(s)	Álvaro Sicilia (FUNITEC), German Nemirovski, Andeas Nolle (HAS)				
Editor	Álvaro Sicilia (FUNITEC)				
DoW	The purpose of Task 4.4 *Interfaces with external tools* is to design and implement interfaces linking the semantic framework with external tools for processing and analysing energy data using web standards, which will enable communication between the semantic framework and different types of applications in a unified manner. The interoperability between different types of tools and the SEIF will be verified in test cases. Operative interfaces with their configuration options and guidelines will be provided for installation. A set of examples showing the interoperability between SEIF and integrated tools will be provided.				
Comments					

Document history			
V	**Date**	**Author**	**Description**
1	2013-09-09	Álvaro Sicilia (FUNITEC)	Preparation of the original draft.
2	2013-09-25	German Nemirovski (HAS)	Inclusion of the Federation engine section.
3	2013-10-05	Alvaro Sicilia (FUNITEC)	Inclusion of the first and second test cases descriptions, as well as the text of the query design guidelines.
4	2013-10-09	German Nemirovski, Andeas Nolle (HAS)	Inclusions of sections Indexing and Looking-up service.
5	2013-10-10	Alvaro Sicilia (FUNITEC)	Inclusion of SPARQL client sections.
6	2013-10-11	German Nemirovski, Andeas Nolle (HAS), Alvaro Sicilia (FUNITEC)	Third test case described. Inclusion of conclusion and summary sections.
7	2013-10-15	Leandro Madrazo (FUNITEC)	Review by project coordinator. This version is submitted to the internal reviewers: Tomas Karlsson (A9), Xavi Cipriano (CIMNE)
8	2013-20-25	Alvaro Sicilia (FUNITEC)	Answers to comments from reviewers

Table of Contents

EXECUTIVE SUMMARY

Deliverable 4.4 *Interfaces with external tools,* developed within Work Package 4 *Semantic energy information framework,* summarises the work done and the results achieved in Task 4.4 *Interfaces with external tools,* whose goal is to design and implement interfaces linking the semantic framework with external tools for processing and analysing energy data using web standards.

Deliverable 4.4 contributes to the project with interfaces –based on standard technologies such as SPARQL– to connect external tools –mainly those developed in WP5– with the data sources collected in WP3 by means of the Semantic Energy Information Framework (SEIF). A federation engine is introduced along with explanations of how interoperability issues (in terms of query language and protocol of access) are solved. The SEIF mediates between tools and data sources at two levels: first, conceptually, by providing a semantic energy model shared by tools and data; and second, technically, through a federation of related data sources and answering data queries.

A set of test cases have been implemented to ensure that the interfaces between the external tools and the SEIF are operative. The test cases cover some of the most relevant scenarios which a tool developer can face.

The report is structured in the following sections:

1. **Introduction**: Purpose of the deliverable, contributions of partners, and relationships between the work undertaken in this task with other work packages.
2. **Interfaces with external tools**: This section describes what is an interface in the context of this project. The purpose of such interfaces is to integrate interoperable tools and the data sources they use by means of a semantic energy model. SPARQL language is introduced and its characteristics are explained in terms of expressivity, flexibility, federation possibilities, and reasoning. Along this line, the federation engine and the indexing and look-up services –which make the server side of the SEIF– are briefly described. Then, two SPARQL clients are described: a PHP library and a RapidMiner operator. Both clients are used in the SEMANCO integrated platform to query the data sources. Finally, a set of guidelines have been collected to support the design of queries. Following these guidelines is important in order to be able to create SPARQL queries with a good performance.
3. **Test cases**: Three test cases have been implemented to verify the interoperability between the different types of tools and the SEIF. The test cases are described using a template. Code snippets and screen captures of the outputs are provided. "Querying the energy model" is the first. Its goal is to verify that the energy model can be queried by an external tool. The second test case, "Querying different sources using the same terminology" checks that data from different sources can be retrieved using the same energy model. This case is relevant insofar as the external tools need to know only one model. Finally, the third test case, "Using different tools to query the SEIF", verifies that different tools can interrogate the SEIF using the same language, infrastructure, and protocol to get the data.
4. **Conclusions**: Contributions of the work carried out in Task 4.4 to the project development, and the impact on other Work Packages and tasks.

1 INTRODUCTION

1.1 Purpose and target group

The purpose of this deliverable is to report about the work done in Task 4.4 *Interfaces with external tools*. The goal of the task has been to implement the interfaces of the SEIF to communicate with external tools –particularly the tools developed in WP5 *Integrated Tools*– by means of Semantic Web technologies. The external tools taken into account are those developed in the project as well as tools from third parties such as RapidMiner.

The main part of the SEIF interface is the central SPARQL endpoint, a web service which accepts queries formulated in the SPARQL Protocol and RDF Query Language (SPARQL) and returns Resource Description Framework (RDF) data. Between the central endpoint and the data sources is the federation engine which send parts of the query to specific sources and merges the results of the different sources into one result set. A set of test cases have been developed to verify the interoperability between the external tools and the SEIF.

Tool developers are the main target group of this document since they have to implement the SPARQL queries to retrieve the data needed by the tools. Furthermore, the maintainers of the integrated platform also belong to the target group of this document.

1.2 Contribution of partners

The work carried out in Task 4.4 was led by FUNITEC. The endpoint has been developed by HAS. The test cases have been implemented by FUNITEC.

1.3 Relations to other activities in the project

The interface implemented and included in the SEIF gives access to the external tools developed in the WP5 *Integrated Tools*. It is an important component of the overall project since it helps to bring together the outputs from WP4 and WP5. Some of the work carried out in the test cases have been used to develop the tools for processing and analysing energy data which are listed below:

1. Energy modelling tools developed in Task 5.1 *Building stock energy modelling tool*.
2. Analysis tools developed in Task 5.2 *Energy analysis, and optimisation and strategic decision tools*.
3. Integrated platform developed in Task 5.4 *Prototype of the integrated platform* and Task 5.6 *Integrated platform.*
4. Semantic data explorer developed in Task 4.3 *User interfaces for knowledge representation*.

Furthermore the work carried out in this task is related to other components of the SEIF, the ontology repository implemented in the Task 3.4 Ontology Repository and Data migration to OWL format.

2 INTERFACES WITH EXTERNAL TOOLS

The SEIF is a mediator between the tools and the data sources insofar as it creates a bridge between different domains by means of the semantic energy model (global ontology) developed in Task 4.2 *Design of a semantic energy model*. Thus, the tools can ask for data from different domains (e.g. building typologies, energy systems, among others) using a common and shared terminology between tools and the integrated data. Since different data sources have been integrated using the semantic energy model –described in D3.*4 Ontology repository with migrated data*– they can be interrogated using that model. This way the SEIF mediates between tools and data sources at two levels: firstly, conceptually, by providing a semantic energy model shared by tools and data; and secondly, technically, by federation of related data sources and answering data queries formulated w.r.t the semantic energy model.

SPARQL is a computer language used to make queries into databases stored in RDF format. It is a W3C recommendation (Prud'hommeaux, 2008) implemented by a large amount of RDF stores providers such as Virtuoso Server, Sesame, AllegroGraph, BigOWLIM, Fuseki, and RDF rewriters such as R2R Server and Ontop. SPARQL has become the standard language of the Semantic Web since it provides flexibility and expressivity to formulate queries, enables data federation, and make possible reasoning tasks over the data. The tools developed in WP5 interrogate the SEIF by means of SPARQL queries which are devised according to the semantic energy model. The tools reach the data in a unified manner thanks to the application of these standard technologies in the whole data retrieval process.

The gateway of the SEIF is a central endpoint, a service which receives the SPARQL queries sent by the external tools using a SPARQL client. The queries are processed by the federation engine which is the core of the SEIF. The output of that process are a set of source-specific queries which are sent to their relative data sources. Each data source sends the response back to the federation engine which harmonises the outputs and sends them back to the client which has launched the initial query (Figure 1).

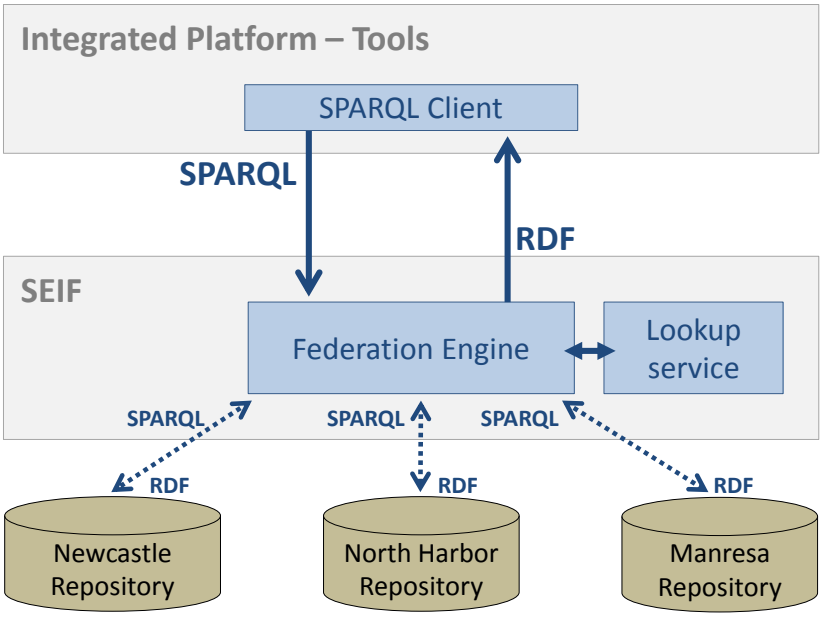

Figure 1. Relations between the Integrated Platform, SEIF, and the repositories

2.1 Federation engine

The federation engine aims at facilitating interoperability of tools with regard to the data they operate on. Thanks to the federation engine, users of the tools formulate queries targeting data distributed over numerous sources as if they would be sent to a single source. The queries neither contain technical details, nor references to the schemas of single sources or are written in a syntax specific to the access methods used by single sources. The only two requirements for the queries are: i) queries should be formulated in standard SPARQL 1.1 (W3C, 2013b) and, ii) they have to use the vocabulary of the energy model.

The SEMANCO federation engine consists of the following components (Figure 2): SPARQL Interface (1), Quest Reasoner (2), Query Joiner (3), Indexing Service (4), Query Optimizer (5) and Executor & Federator (6).

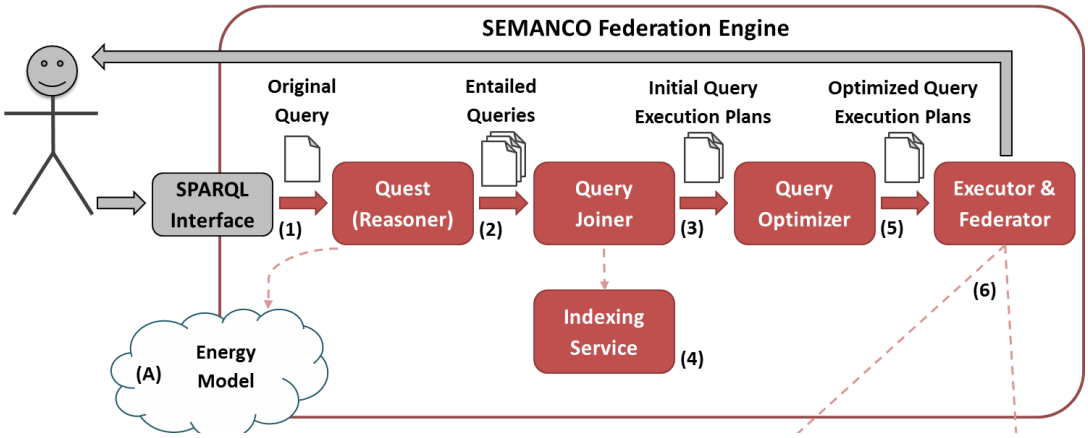

Figure 2. System Architecture of the Semantic Energy Information Framework (Nolle et al., 2013)

The roles of these components in the query processing is described next:

1. A SPARQL query is formulated in terms of the energy model and launched by client-tools at the central SPARQL endpoint interface (1).
2. The query is rewritten using Quest reasoner (2) that applies inference rules based on the RDFS and DL-Lite$_{\mathcal{A}}$ entailment regimes. The goal is to generate all possible –with regard to the energy model– multiple queries expressing semantically the same question as the one initially launched by an external tool.
3. At the next steps each query previously generated is subdivided into its elementary parts, so called basic graph patterns (BGP). This is done by the Query Joiner (3). Each of the BGPs is then evaluated by means of the indexing service (4). As a result, selected BGPs are associated with the sources that –according to the statistic information collected in the index– may contain data relevant for these BGPs.
4. In the next step, the query optimizer (5) generates an execution plan. It defines the querying sequence for the single sources. The idea behind the execution plan is to achieve the maximum efficiency while querying single sources. The optimisation of the execution plan aims at minimising the data transfer and the overall query execution time that often depends on the sequence of querying single data sources.
5. Executer and Federator (6) forwards queries consisting of BGPs selected after the index look-up (step 3) to the SPARQL endpoints of single sources according to the execution plan. The query answers are processed and then the answer for the query initially launched by an external tool (step 1) is generated.

2.2 Indexing and look-up service

Since data is distributed over the sources integrated into the SEMANCO repositories we assume that a single source is only able to answer a part of a query launched by a tool or a user. Hence, it does not make sense to forward the query as it is to a source that very probably would return an empty answer. In this context the task of the federation engine is to identify relations between data sources and a single query or "graph" patterns (BGPs) contained in the original query in order to generate sub-queries that can be answered by single sources. For this purpose, an index catalogue containing statistics for single sources and an index look-up service is required.

The index catalogue implemented as a part of SEMANCO federation engine extends the QTree approach developed by (Harth et al., 2010) that in turn is based on the R-Tree structure. (Guttman, 1984). The idea of the approach described in Deliverable 4.5 *Semantic Energy Information Framework* (Nolle et al., 2013) and in (Nolle & Nemirovski, 2013) is to summarise data stored in single data sources. With this purpose, all triples stored in single integrated sources are retrieved by querying of SPARQL endpoints of each source. The triples are indexed by means of hash values which are calculated for all RDF-triple elements such as the subject, object and predicate. Since hash values are figures, mathematical comparisons (e.g. greater/less) can be used to group them in so-called Minimum Bounding Boxes (MBBs) which are associated with a certain data source.

Invoking the index look-up service for a BGP leads to selections of MBBs for each source containing RDF triples that satisfy this pattern. For this purpose, the corresponding hash values are calculated for each pattern elements defined by a URI.

In order to fill the index catalogue, the indexing service interrogates the SPARQL endpoints of single sources. The interaction is initiated by the indexing service that –in this case– performs the client role while the SPARQL endpoints act as servers.

2.3 SPARQL client

A SPARQL client is an application which encapsulates the methods required to interrogate SPARQL endpoints. Those methods implement the SPARQL communication protocol between the client and the endpoint (W3C, 2013a). When HTTP is used as transportation protocol the queries can be sent through GET or POST operations. Numerous libraries for different programming languages such as Javascript, Java, PHP, and Python implementing SPARQL client methods are available. The official list can be found in W3C (2013c).

2.3.1 PHP client

The tools developed in SEMANCO –embedded, interfaced, external– which are further integrated in the platform interact with the SEIF by means of a SPARQL client. Most of the tools and the platform are primarily developed in PHP language. In the research community the following libraries for PHP are available: ARC2[1], EasyRDF[2], RDF API for PHP[3], and Graphite PHP Linked Data Library[4]. All of these libraries provide a SPARQL client, serialise the most widely used formats such as RDF/XML, RDF/JSON, N-Triples, among others, are open-source and easy to use.

The following example shows how to send a query to a SPARQL endpoint using a ARC2 library:

[1] https://github.com/semsol/arc2
[2] http://www.easyrdf.org
[3] http://wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/
[4] http://graphite.ecs.soton.ac.uk/

```
1.  include_once('path/to/arc/ARC2.php');
2.  $store = ARC2::getRemoteStore(array('remote_store_endpoint' =>
    'http://example.com/sparql'));
3.  $q = 'PREFIX sumo: <http://www.ontologyportal.org/SUMO.owl#>
          SELECT ?building WHERE {
             ?building a sumo:Building.
          }';
4.  $rows = $store->query($q, 'rows');
5.  $first_Building = $rows[0]['building'];
```

The ARC2 library is included in the first line of the code. Then, in the second line, the endpoint is configured. In this example only the URL address of the endpoint is set. However other parameters can be set too, such as the endpoint timeout. In the third line, the variable *$q* contains a SPARQL query to retrieve all instances whose class are buildings. The method *$store->query* sends the query to the endpoint and the result is stored in the *$rows* array. In the fifth line it is shown how the *$rows* array is accessed. The first dimension of the array contains the result set, and the second dimension contains the variables of the SPARQL query. In the example the variable *$first_Building* is set to the variable *building* retrieved from the endpoint.

2.3.2 RapidMiner operator

Within Task 5.2 *Energy analysis, and optimization and strategic decision tools* (Nemirovski et al., 2013) an extension of a data mining tool, RapidMiner, has been developed. This development was motivated by the need to integrate the tool into the data analysis process carried out in the SEMANCO platform, primarily by enabling it to retrieve data stored in data sources integrated into the SEIF repository. In this context the tool acts as a SPARQL client and therefore needs a SPARQL client interface.

Such an interface has been implemented as a new RapidMiner operator that is de facto an extension of the RapidMiner environment. The user can find the operator in the operators' palette, drag it and drop it onto the working desk and connect it with other operators to a flow (Figure 3).

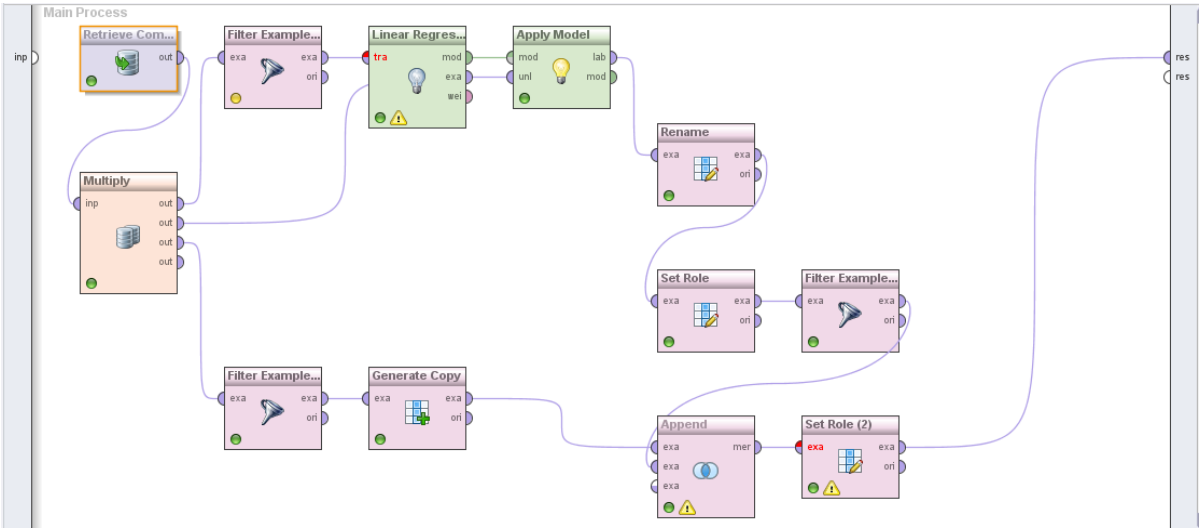

Figure 3. Working desk of RapidMiner showing a data mining process containing multiple operators (see also Deliverable 5.2, Figure 11)

The operator accepts a SPARQL query as a parameter, submits it to the SEIF endpoint, receives the RDF data and converts it into a RapidMiner native representation of RapidMiner. After this is done, Data Mining operators are available in the standard edition of the tool

which can be used to process the data. The conversion process is highly automated. The data types query result variables are automatically transformed into the corresponding RapidMiner type. For example, URIs of individuals are transformed into strings and Boolean variables are transformed into binomials.

The operator to perform data retrieval is placed at the beginning of the data mining process flow. When data is retrieved in the RDF format and then translated into the native format of RapidMiner it is processed by all subsequent operators in the flow. This way, the operators can work with this data in the same way as it would have been retrieved form a relational database or an Excel table.

2.4 Guidelines

The SEIF is giving access to more than 3 million triples distributed in the three data repositories and referring to more than 70 concepts of the energy model. The architecture of the repositories –explained in D3.4 *Ontology repository with migrated data*– is based on RDF rewriters such as D2R Server. Those kind of rewriters translate SPARQL queries into SQL without perform –in query execution– as well as relational databases queried by native SQL (Bizer and Schultz, 2009). However, the current research in this area has produced significant advances such as Ultrawrap whose performance is close enough to traditional databases (Sequeda and Miranker, 2012).

With this limitation in mind, it is important to design SPARQL queries that once translated into SQL can perform well. One of the main difficulties to overcome are queries that generate very large intermediate results such as queries containing non-selective graph patterns. Salvadores et al. (2012) has collected a set of recommendations which can be applied in the design of the queries to avoid these kinds of problems:

1. It is good practice to query the endpoint iteratively with selective queries instead of a big non-selective query. This way the client should manage the query calls and process the outputs.
2. It is recommended to use Offset and Limit operators to reduce the data moved between the endpoint and the client.
3. It is necessary to be aware of the use of Group by and Order by since the query engine needs to aggregate or sort all query solutions. That is, the execution time can grow until the timeout is hit.
4. It is recommendable to implement a cache system in the client side since it cannot be expected to get a response within a small time range. Thus, the repetitive queries can be cached in order to avoid sending queries to which the response is already known.

An illustrative example of the first recommendation in the context of the SEMANCO project would be a process that retrieves the space heating and cooling values of specific buildings. Accordingly, the Query 1 should be avoided since it generates inefficient SQL statements; in the case of D2R Server is generating more than 400 unions which are not processable by a conventional database such as MySQL.

Query 1. Inefficient SPARQL query

```
PREFIX semanco: <http://www.semanco-project.eu/2012/5/SEMANCO.owl#>
SELECT DISTINCT ?value heating ?value cooling
WHERE {
    ?instance_building semanco:hasBuilding_Use _:Residential .
    ?instance building semanco:hasAge [semanco:hasFrom Year [semanco:fromYearValue
?value_fromyear]].
```

```
    ?instance building
semanco:hasEnergy Consumption And Energy Saving Related To Building Services
[semanco:hasEnergy Quantity And Emission ?instance quantity].
    ?instance quantity semanco:hasEnergy Service ?instance service1 .
    ?instance_service1 a semanco:Space_Heating ;
          semanco:delivered_EnergyValue ?value_heating .
    ?instance quantity semanco:hasEnergy Service ?instance service2 .
    ?instance service2 a semanco:Space Cooling ;
          semanco:delivered EnergyValue ?value cooling .
    FILTER ( ?value_fromyear < 2020 ) .
}
```

To solve the performance problem, Query 1 can be separated into three queries. However, the client application should transfer parameters between the subqueries. First the client would call Query 2 to get a residential building which has been built before the year 2020.

Query 2. SPARQL Query to retrieve residential building built before year 2020

```
PREFIX semanco: <http://www.semanco-project.eu/2012/5/SEMANCO.owl#>
SELECT DISTINCT ?instance building
WHERE {
    ?instance building semanco:hasBuilding Use  :Residential ;
          semanco:hasAge [ semanco:hasFrom_Year [semanco:fromYearValue ?value_fromyear]].
    FILTER( ?value_fromyear < 2020 ) .
}
```

Then the following two queries (Query 3 and 4) would be launched to get the space cooling and heating values. The client application would generate both queries using the output of the previous query.

Query 3. SPARQL query to retrieve the space heating value of a specific building

```
PREFIX semanco: <http://www.semanco-project.eu/2012/5/SEMANCO.owl#>
SELECT DISTINCT ?value_heating
WHERE {
    <'.$instance building.'>
semanco:hasEnergy Consumption And Energy Saving Related To Building Services
[semanco:hasEnergy Quantity And Emission [semanco:hasEnergy Service ?instance service] ].
    ?instance_service a semanco:Space_Heating ;
          semanco:delivered_EnergyValue ?value_heating .
}
```

Query 4. SPARQL query to retrieve the space cooling value of a specific building

```
PREFIX semanco: <http://www.semanco-project.eu/2012/5/SEMANCO.owl#>
SELECT DISTINCT ?value cooling
WHERE {
    <'.$instance_building.'>
semanco:hasEnergy_Consumption_And_Energy_Saving_Related_To_Building_Services
[semanco:hasEnergy Quantity And Emission [semanco:hasEnergy Service ?instance service] ].
    ?instance service a semanco:Space Cooling ;
          semanco:delivered EnergyValue ?value cooling .
}
```

Following the fourth recommendation, this process can be improved if the intermediate results are cached. So if there are many buildings with the same use and year of construction then some queries can be avoided and the performance improved.

Tool developers should take into account the previous guidelines in order to build high-performance tools that take advantage of the SEIF endpoint. These guidelines are also available –as an online help– in the SEIF endpoint web site to inform external people of best practice.

3 TEST CASES

Three test cases have been implemented to verify the interoperability between different types of tools and the SEIF. That is, the different tools can interrogate the SEIF in a unified manner and can get data from the different sources. Each test case is described using the following template and also is included as an excerpt of the source code that implements the test case.

Purpose	Goal of the test case.
Description	Description of the test case and its context of execution.
Requirements	Requirements to be fulfilled before carry out the test case.
Action	Explanation of the action.
Expected results	Results that should be obtained if the test case works.
Actual Result	Results obtained in the test case execution.
Comments	Remarks regarding the test case.
Pass/Fail	Weather the results of the test case are the expected ones or not.

The implementation of the test cases has enabled the verification of the appropriateness of the interfaces between the external tools and the SEIF.

3.1 Test case 1. Querying the energy model

The purpose of test case 1 is to ensure that the energy model can be queried by an external tool. An example of an external tool could be the Semantic Data Explorer described in D4.3 *User interfaces for domain experts interacting with SEIF* which uses the energy model (TBox) to guide the exploration process. The test case is defined following the template (Test case 1).

Test case 1. Query the energy model (TBox only)

Purpose	Verify that the tools can reach the ontology classes and properties through the SEIF to use it in the interface.
Description	In the scenario of the test case, the tool needs to get a list of the building uses and sub uses to populate a select box where the user of tool can select a use of the building. The building uses have been defined in the Standard Tables which are coded in the global ontology.
Requirements	The pre-condition is to run the SEIF with the data repositories and the semantic energy model (global ontology)
Actions	Retrieve and display the building uses.
Expected results	The list of the building uses of the ontology.
Actual Result	The list of the building uses of the ontology.
Comments	This test case can be extrapolated to retrieve any ontology structure needed by the tools. The results have been successful.
Pass/Fail	Pass.

Source code 1 is a PHP snippet which implements test case 1. The ARC2 library is included in the first line . In the second line is the remote store with the URL of the endpoint which currently is: http://semanco01.hs-albsig.de/ELITE/sparql. Then, the variable *$q* is set with a

SPARQL query to retrieve the building uses and subuses ordered by the building use. In the fourth line, the query is sent to the endpoint and the result is stored in the *$building_uses* array. Finally in the fifth and sixth lines the building uses and subuses are displayed.

Source code 1. Code to retrieve the building uses from the SEIF and display them.

```
1.  include_once('path/to/arc/ARC2.php');
2.  $store = ARC2::getRemoteStore(array('remote_store_endpoint' => 'http://semanco01.hs-
    albsig.de/ELITE/sparql'));
3.  $q = 'PREFIX semanco: <http://www.semanco-project.eu/2012/5/SEMANCO.owl#>
          PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
          SELECT DISTINCT ?BUse ?BSubUse
          WHERE {
             ?BUse rdfs:subClassOf semanco:Building Use .
             ?BSubUse rdfs:subClassOf ?BUse .
          } ORDER BY ?BUse';
4.  $building_uses = $store->query($q, 'rows');
5.  foreach($building_uses as $building_use){
6.     echo 'Uses: '.$building use['BUse'].' -> '. $building use['BSubUse'] . '<br>';
7.  }
```

This test case demonstrates that an external application can query the ontology using SPARQL queries.

3.2 Test case 2. Querying different sources using the same terminology

The purpose of test case 2 is to retrieve data from different data sources using the same energy model (terminology and relations). That means that the external tool only needs to know one model (the global ontology) to query all data sources. This is an advantage if we compare it to other contexts –for example in Linked Open Data environments– where the tool would need to know the schemas of each dataset queried. These kinds of queries are used in the integrated platform.

Test case 2. Query different sources with the same ontology

Purpose	Verify that the interfaces implemented solve the interoperability between different data sources.
Description	In the scenario of the test case, the tool needs to retrieve a list of the building typologies –based on year of construction– from the data sources.
Requirements	SEIF running with the data repositories and the semantic energy model (global ontology)
Actions	Retrieve and display the building typologies grouped by municipality.
Expected results	A list of building typologies for each municipality.
Actual Result	A list of building typologies for each municipality.
Comments	There are only two municipalities which have building typologies (Manresa and Copenhagen).
Pass/Fail	Pass.

Source code 2 is a PHP snippet which implements test case 2. As the previous snippets, in the first two lines the endpoint is set. Then, the variable *$q* is set with a SPARQL query to retrieve the age class of the building typologies of the municipalities ordered by the municipality and starting year (fromYear). In the fourth line, the query is sent to the endpoint and the result is stored in the *$mlist* array. Finally in the fifth and sixth lines are the contents of the array: municipality, age class (from and to year) and number of buildings which have that age class.

Source code 2. Code to retrieve the year construction list from the SEIF and display them.

```
1.  include_once('path/to/arc/ARC2.php');
2.  $store = ARC2::getRemoteStore(array('remote_store_endpoint' => 'http://semanco01.hs-
    albsig.de/ELITE/sparql'));
3.  $q = 'PREFIX sumo: <http://www.ontologyportal.org/SUMO.owl#>
          PREFIX semanco: <http://www.semanco-project.eu/2012/5/SEMANCO.owl#>
          SELECT distinct ?municipality ?fromYear ?toYear count(?b) AS ?num WHERE {

            ?municipality semanco:hasNeighbourhood [semanco:hasLand [semanco:hasBuilding
?b]].
            ?b a sumo:Building;
               semanco:hasAge ?age.

            ?age semanco:hasFrom_Year [semanco:fromYearValue ?fromYear];
               semanco:hasTo Year [semanco:toYearValue ?toYear].

          } group by ?municipality ?fromYear ?toYear order by ?municipality ?fromYear';
4.  $mlist = $store->query($q, 'rows');
5.  echo '<table><tr><td>municipality</td><td>fromYear</td><td>toYear</td><td>num</td</tr>
    ';
6.  foreach($mlist as $mun){
7.        echo '<tr><td>'.$mun[municipality].'</td><td>'. $mun['fromYear'] . '</td><td>'.
    $mun['toYear'] . '</td><td>'. $mun['num'] . '</td></tr>';
8.  }
9.  echo '</table>';
```

The output of source code 2 is a table with four columns: municipality, fromYear, to Year, and number of buildings (Figure 4).

municipality	fromYear	toYear	num
http://www.semanco-project.eu/municipality/Copenhagen	-	1900	50
http://www.semanco-project.eu/municipality/Copenhagen	-	1930	50
http://www.semanco-project.eu/municipality/Copenhagen	-	1980	50
http://www.semanco-project.eu/municipality/Copenhagen	-	2007	50
http://www.semanco-project.eu/municipality/Copenhagen	1901	1940	50
http://www.semanco-project.eu/municipality/Copenhagen	1901	1950	50
http://www.semanco-project.eu/municipality/Copenhagen	1941	1960	50
http://www.semanco-project.eu/municipality/Copenhagen	1941	1960	50
http://www.semanco-project.eu/municipality/Copenhagen	1961	1980	50
http://www.semanco-project.eu/municipality/Copenhagen	1961	1972	50
http://www.semanco-project.eu/municipality/Copenhagen	1981	2007	50
http://www.semanco-project.eu/municipality/Copenhagen	1981	1978	50
http://www.semanco-project.eu/municipality/Copenhagen	2008	-	50
http://www.semanco-project.eu/municipality/Copenhagen	2008	1998	50
http://www.semanco-project.eu/municipality/Manresa	1901	1940	1
http://www.semanco-project.eu/municipality/Manresa	1901	1950	1
http://www.semanco-project.eu/municipality/Manresa	1941	1960	1
http://www.semanco-project.eu/municipality/Manresa	1941	1960	1
http://www.semanco-project.eu/municipality/Manresa	1961	1980	1
http://www.semanco-project.eu/municipality/Manresa	1961	1972	1
http://www.semanco-project.eu/municipality/Manresa	1981	2007	1
http://www.semanco-project.eu/municipality/Manresa	1981	1978	1
http://www.semanco-project.eu/municipality/Manresa	1981	1993	1
http://www.semanco-project.eu/municipality/Manresa	1981	2013	1
http://www.semanco-project.eu/municipality/Manresa	1994	2005	1
http://www.semanco-project.eu/municipality/Manresa	2006	-	1
http://www.semanco-project.eu/municipality/Manresa	2008	-	1

Figure 4. Outputs of the source code 2.

3.3 Test case 3. Using different tools to query the SEIF

The purpose of test case 3 is to validate that different tools can interrogate the SEIF using the same model. That means that there is no specific way to retrieve the data depending on the tool used. That is, all external tools use the same language, infrastructure and protocol to obtain the data. In test case 3, two different tools query the SEIF to retrieve data.

Test case 2. Query different sources with the same ontology

Purpose	Verify that the interfaces implemented solve the interoperability between different tools and the SEIF.
Description	In the scenario of the test case, there are two tools that will retrieve a list of the building typologies –based on year of construction– and their energy used for space heating from the data sources. Both tools should use the same query language, infrastructure and protocol to get the data.
Requirements	SEIF running with the data repositories and the semantic energy model (global ontology)
Actions	1. Retrieve and display the building typologies and their energy heating values. 2. Retrieve the same data from Rapidminer tool.
Expected results	A list of building typologies and their energy heating values.
Actual Result	A list of building typologies and their energy heating values.

Comments	-
Pass/Fail	Pass.

Source code 3 is a PHP snippet which implements test case 3. As the previous snippets, in the first two lines the endpoint is set. Then, the variable *$q* is set with a SPARQL query to retrieve the age class of the building typologies and their energy values for space heating. In the fourth line, the query is sent to the endpoint and the result is stored in the *$rows* array. Finally in the fifth to ninth lines are the contents of the array: municipality, age class (from and to year) and number of buildings which have that age class.

Source code 3. Code to retrieve the year construction list from the SEIF and display them.

```
1.  include_once('path/to/arc/ARC2.php');
2.  $store = ARC2::getRemoteStore(array('remote_store_endpoint' => 'http://semanco01.hs-
    albsig.de/ELITE/sparql'));
3.  $q = 'PREFIX sumo: <http://www.ontologyportal.org/SUMO.owl#>
          PREFIX semanco: <http://www.semanco-project.eu/2012/5/SEMANCO.owl#>
          SELECT distinct ?b ?fromYear ?toYear ?value_heating WHERE {

          <http://www.semanco-project.eu/municipality/Copenhagen> semanco:hasNeighbourhood
[semanco:hasLand [semanco:hasBuilding ?b]].
          ?b a sumo:Building;
                semanco:hasAge ?age.
          ?age semanco:hasFrom_Year [semanco:fromYearValue ?fromYear];
                semanco:hasTo_Year [semanco:toYearValue ?toYear].

          ?b semanco:hasEnergy Consumption And Energy Saving Related To Building Services
[semanco:hasEnergy Quantity And Emission [semanco:hasEnergy Service ?instance service] ].
          ?instance service a semanco:Space Heating ;
                semanco:delivered_EnergyValue ?value_heating .
          }';
4.  $rows = $store->query($q, 'rows');
5.  echo '<table><tr><td>b</td><td>fromYear</td><td>toYear</td><td>value heating</td</tr>
    ';
6.  foreach($rows as $r){
7.        echo '<tr><td>'.$r[b].'</td><td>'. $r['fromYear'] . '</td><td>'. $r['toYear'] .
    '</td><td>'. $r['value heating'] . '</td></tr>';
8.  }
9.  echo '</table>';
```

The output of source code 3 is a table with four columns: building, fromYear, to Year, and energy heating value (Figure 5).

b	fromYear	toYear	value_heating
http://www.semanco-project.eu/building/NHD179	-	1900	133
http://www.semanco-project.eu/building/NHD4	-	1900	178
http://www.semanco-project.eu/building/NHD182	-	1900	197
http://www.semanco-project.eu/building/NHD7	-	1900	238
http://www.semanco-project.eu/building/NHD185	-	1900	109
http://www.semanco-project.eu/building/NHD186	-	1900	109
http://www.semanco-project.eu/building/NHD12	-	1900	134
http://www.semanco-project.eu/building/NHD187	-	1900	109
http://www.semanco-project.eu/building/NHD188	-	1900	110
http://www.semanco-project.eu/building/NHD189	-	1900	194
http://www.semanco-project.eu/building/NHD190	-	1900	99
http://www.semanco-project.eu/building/NHD18	-	1900	225
http://www.semanco-project.eu/building/NHD193		1900	181

Figure 5. *Outputs of the source code 3.*

The second tool involved in the test case is RapidMiner. As already described in Section **¡Error! No se encuentra el origen de la referencia.** we have developed a RapidMiner operator that - though generally applicable - serves in the project SEMANCO for interaction with the SPARQL end-points of the SEIF. A user who applies the operator, generates a SPARQL query that refers to the terms of the energy model. The technology applied for the query generation has no impact on the functioning of the operator. Such a query can be formulated manually. Alternatively, one of the automation instruments described in Deliverable 5.2 *Tools for energy analysis* can be applied. The latter approach makes sense especially for big Queries, for example, those with more than 100 characters.

In our example we use the same query as the one used in the source code 3 shown above. This query is submitted to the SEIF endpoint using the GUI of the RapidMiner operator mentioned above (Figure 6).

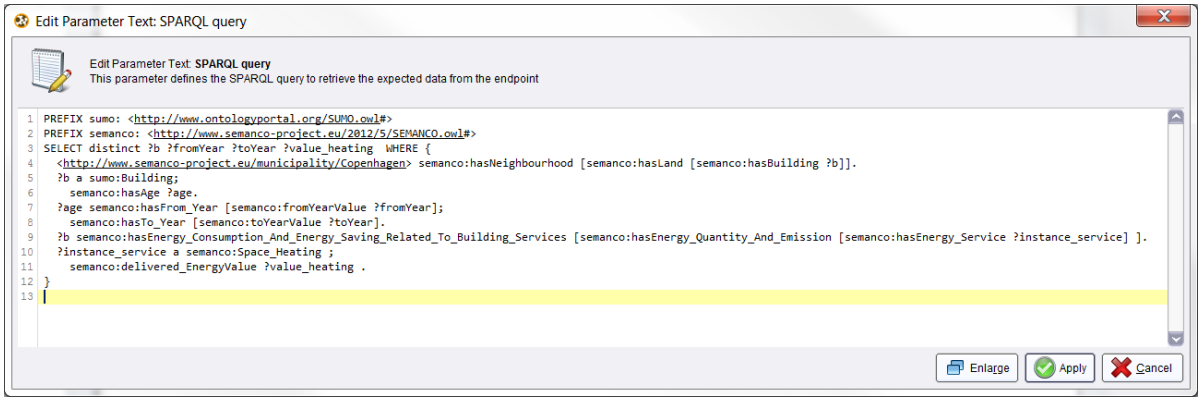

Figure 6. Input parameters for the RDF retrieval operator

The query answer provided by the SEIF contains data in RDF format that has been retrieved from all data sources integrated into the SEMANCO repositories. The operator converts the retrieved data into the native rapid miner representation, that is based upon a relational schema which contains – apart of the retrieved values – a bunch of metadata required by the various data mining algorithms included in the RapidMiner environment. The retrieved data represented as so called RapidMiner Example Set is shown in Figure 7. Metadata describing the example set are shown in Figure 8.

ExampleSet (1416 examples, 1 special attribute, 4 regular attributes)

ID	b	fromYear	toYear	value_heating
0	http://www.semanco-project.eu/building/ED1	2010	2015	29.880
1	http://www.semanco-project.eu/building/ED10	2021	2030	8.400
2	http://www.semanco-project.eu/building/ED11	2021	2030	5.900
3	http://www.semanco-project.eu/building/ED12	2021	2030	5.900
4	http://www.semanco-project.eu/building/ED13	2031	2050	8.400
5	http://www.semanco-project.eu/building/ED14	2031	2050	8.400
6	http://www.semanco-project.eu/building/ED15	2031	2050	5.900
7	http://www.semanco-project.eu/building/ED16	2031	2050	5.900
8	http://www.semanco-project.eu/building/ED2	2010	2015	29.880
9	http://www.semanco-project.eu/building/ED3	2010	2015	32.200
10	http://www.semanco-project.eu/building/ED4	2010	2015	32.200
11	http://www.semanco-project.eu/building/ED5	2016	2020	12.400
12	http://www.semanco-project.eu/building/ED6	2016	2020	12.400
13	http://www.semanco-project.eu/building/ED7	2016	2020	12.900
14	http://www.semanco-project.eu/building/ED8	2016	2020	12.900
15	http://www.semanco-project.eu/building/ED9	2021	2030	8.400
16	http://www.semanco-project.eu/building/NHD1	?	1930	221
17	http://www.semanco-project.eu/building/NHD1	0	1930	221
18	http://www.semanco-project.eu/building/NHD1	?	1900	221
19	http://www.semanco-project.eu/building/NHD1	0	1900	221
20	http://www.semanco-project.eu/building/NHD10	?	1930	130
21	http://www.semanco-project.eu/building/NHD10	0	1930	130
22	http://www.semanco-project.eu/building/NHD10	?	1900	130
23	http://www.semanco-project.eu/building/NHD10	0	1900	130
24	http://www.semanco-project.eu/building/NHD100	1961	1972	52
25	http://www.semanco-project.eu/building/NHD100	1961	1972	52
26	http://www.semanco-project.eu/building/NHD100	1961	1980	52

Figure 7. Retrieved data transformed in RapidMiner representation format

ExampleSet (1416 examples, 1 special attribute, 4 regular attributes)

Role	Name	Type	Missings
id	ID	integer	0
regular	b	text	0
regular	fromYear	integer	200
regular	toYear	integer	100
regular	value_heating	real	0

Figure 8. Meta data of the retrieved data transformed in RapidMiner representation format

4 CONCLUSIONS

4.1 Contribution to overall picture

The interfaces implemented in Task 4.4 *Interfaces with external tools* and described in this deliverable are the glue which connects the external tools and the SEIF. The interoperability problems between tools and data are avoided because the interfaces are based on the same language (SPARQL), infrastructure (a central endpoint), and protocol (HTTP) which are standard technologies of the Semantic Web. SPARQL is the standard language to query semantic data –specifically RDF data– providing the necessary expressivity and reasoning capabilities. The queries are sent to the SEIF through its central endpoint, a web service based on the HTTP protocol. The use of Semantic Web standards to implement the SEIF interfaces makes it unnecessary for the tools to use different models, protocols, interfaces and technologies to access the data. In fact, they only need to know the semantics behind the data which is coded in an ontology and the SPARQL language.

A federation engine has been developed to work as a mediator between the data sources and the external tools. The external tools see the SEIF endpoint as a single source whereas the data sources are integrated by means of the federation engine. The indexing and look-up services are two important components which are necessary to optimise the query generation.

Two SPARQL client applications have been presented. A PHP client is used in the integrated platform as well as in some of the tools developed in WP5. A RapidMiner operator developed in the project is used to query SPARQL endpoints and to transform the output to RapidMiner types.

A set of recommendations have been included in the document to help tool developers to design efficient queries. Good performance queries are needed since the SPARQL-SQL rewrites are not mature enough.

The three test cases implemented demonstrate that the interoperability between external tools and data sources is solved through a unique model and interface (i.e. SPARQL endpoint). This way, the tools can query any data sources integrated by the SEIF.

4.2 Impact on other WPs and Tasks

The work done in this task is related to the Task 4.2 *Design of a semantic energy model* which describes the process of creation of the SEMANCO ontology. Task 4.5 *Semantic energy information framework integration* which integrates all the work carried out in WP4 including the federation engine is also closely related to this task.

On the other hand, the SPARQL client presented in this document has been used in the tools and in the integrated platform developed in WP5 to query the SEIF. Furthermore, the guidelines listed in this document have been applied by the tool developers in the developments carried out in WP5.

4.3 Contribution to demonstration

The work carried out in this task has no direct contribution to the demonstaration. However, it contributes to facilitate the connections between the tools and the integrated platform with the SEIF and the data sources.

The interfaces presented in this document are necessary for the tools developed in WP5 to reach the data sources using the semantic energy model.

4.4 Other conclusions and lessons learned

The work carried out is based on standards widely used in the Semantic Web community. During the implementation of the test cases some issues were raised regarding the query response performance. This led to the project devising some guidelines to support the query generations. The guidelines have been contrasted with other reference works found in the literature.

5 REFERENCES

Bizer, C., & Schultz, A. (2009) The berlin sparql benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)* 5.2: 1-24.

Guttman, A. (1984). R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data* (pp. 47-57). ACM.

Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.-U., & Umbrich, J. (2010). Data summaries for on-demand queries over linked data. In *Proceedings of the 19th international conference on World wide web* (pp. 411-420). ACM.

Nemirovski, G., Nolle, A., Wolters, M., Sicilia, Á., & Madrazo, L. (2013) Deliverable 5.2: Tools for Energy Analysis. In *SEMANCO*: http://semanco-project.eu/index_htm_files/SEMANCO_D5.2_20130731.pdf

Nolle, A, Nemirovski, G. & Sicilia, Á. (2013). Deliverable 4.5: Semantic Energy Information Framework. In *SEMANCO*.

Nolle, A., & Nemirovski, G. (2013). ELITE: An Entailment-based Federated Query Engine for Complete and Transparent Semantic Data Integration. In *Proceedings of the 26th International Workshop on Description Logics.*

Prud'hommeaux, E. & Seaborne, A. (2008) SPARQL Query Language for RDF. *W3C Recommendation*.

Salvadores, M., Horridge, M., Alexander, P.R., Fergerson, R.W., Musen, M.A. & Noy, N.F. (2012) Using SPARQL to query bioportal ontologies and metadata. In *The Semantic Web–ISWC 2012*, pp. 180-195. Springer Berlin Heidelberg.

Sequeda, J.F., & Miranker, D. P. (2012) Ultrawrap: Sparql execution on relational data. *Technical Report TR-12-10*, University of Texas at Austin, Department of Computer Sciences.

World Wide Web Consortium (2013a) SPARQL 1.1 Protocol. http://www.w3.org/TR/sparql11-protocol/ (Accessed: 10/10/2013).

World Wide Web Consortium (2013b) SPARQL 1.1 Query Language. http://www.w3.org/TR/sparql11-query/ (Accessed: 10/10/2013).

World Wide Web Consortium (2013c) SparqlImplementations – W3C Wiki. http://www.w3.org/wiki/SparqlImplementations#Client_Side (Accessed: 10/10/2013).