Interactive Framework for Visual Exploratory Search and Integration of Semantic Web contents and services

Bin Piao and Yuzuru Tanaka
Meme Media Laboratory
Hokkaido University
Sapporo, Japan
e-mail: {piaobin,tanaka}@meme.hokudai.ac.jp

***Abstract*— In this paper, we propose a new interactive framework for Exploratory Search and Visualization of Semantic Web resources, and extend this framework to search and invoke Semantic Web Services. This framework enables users to dynamically extract and expand local schemata of the RDF data in an exploratory way starting from one or more classes of arbitrarily chosen Semantic Web resource. These schemata are generated either by logically combining different classes, or by combining different classes using the logical operators. Using these dynamically extracted schemata, users can visually specify queries and search the target Semantic Web resources for the information of their interest. They can dynamically repeat the process of schema extraction, its expansion, visual query definition, and query evaluation for obtaining the result visualization in an exploratory way until they find all the information they want to obtain. Users can also integrate the search results from different schemata by directly manipulating these schemata, or integrate the resources between Semantic Web Services and normal RDF datasets by directly connecting the retrieved Sematic Web Services to the schemata of the normal RDF datasets.**

Keywords-semantic web; semantic web service; exploratory search; visual query definition; information visualization; information integration

I. INTRODUCTION

In recent years, more than 300 RDF [1] datasets have been published on the web, and can be queried using the RDF query language SPARQL [2]. For creating a SPARQL query, the end users have to be fully-aware of the schemata of target RDF data, which is difficult even for experts. An effective approach for obtaining the schemata of RDF data is to dynamically extract and expand local schemata of the RDF data in an exploratory way starting from one or more classes of arbitrarily chosen Semantic Web resource. Even if the users know the schemata of these RDF datasets, it is also difficult to define a complex SPARQL query, such as using the logical operations, without prior knowledge about SPARQL.

In addition, because of the increasing number of RDF datasets published on the web, integrating different data sources is becoming more and more important. In the dataset integration, the sources of semantic web services (SWSs) can't be ignored. Sometimes we need to integrate the sources of SWSs and normal RDF datasets to obtain users' interested information. For example, users may want to find out some SWSs that can retrieve music events in a specified place, and to search the information about the artists of some selected music events from RDF datasets such as DBpedia. In the Semantic Web, there are some SWS repositories such as iServe [3] for users to choose SWSs according to their purposes. Since the descriptions of the SWSs in these repositories are described by RDF, these SWSs can be queried by SPARQL.

In this paper, we propose a new interactive visual framework for Exploratory Search and Visualization of Semantic Web resources (ESVSW). The ESVSW framework provides a visual component called Class Component to represent the classes in the schemata of the RDF data. Users can directly manipulate these class components to sequentially extract their related classes either by using the sematic relationships among the instances of classes, or by combining different classes using the logical operators. Users can also visually specify queries and search the target Semantic Web resources for their interested information by directly manipulating specific class components. Users can also connect specific class components of these schemata according to the semantic relationships among their classes for integrating the search results from different schemata.

Users can use some visualization components provided by this ESVSW framework to visualize the obtained instances of some class, only through directly connecting a visualization component to the class component. Users can directly select some interested parts of the generated visualization and reuse them to rewrite the queries.

In addition, we extend the ESVSW framework to enable users to search and invoke SWS. The extended ESVSW framework provides a visual component called the service component, to represent the APIs of the retrieved SWSs. In the extended ESVSW framework, Users can directly combine different service components to generate composite SWSs, or to directly connect service components to class components to integrate the sources of some SWSs and RDF datasets in the same work space.

This paper is organized as follows. Section II compares our framework with other related works. Section III represents the ESVSW framework. Section IV represents the extended ESVSW framework.

978-1-4799-4746-1/14/$31.00©2014 IEEE

II. RELATED WORK

In this section we will give an overview about different approaches to exploratively search and integrate the Semantic Web contents, and compare them with our ESVSW framework.

iSPARQL [4] helps users to define SPARQL queries by providing a graph interface. But its users have to understand the schema for defining a SPARQL query. Mashpoint [5] provides some web applications to search and visualize the information from the multiple RDF datasets. It is limited only to search and visualize the RDF data using provided applications. gFacet [6], mspace [7], and Humboldt [8] propose an approach called "facet searching" for exploratively searching the Semantic Web resources. These systems enable users to interactively extract the schema of the target RDF data, and to filter the results by directly selecting the items on these multiple facets. But these systems can generate SPARQL queries in which no logical operator is used. Furthermore, these systems cannot integrate different data sources. MashQL [9] is a semantic version of Yahoo! Pipes. It enables users to exploratively search and integrate the semantic web resources. But it can generate only such SPARQL queries in which no logical operator is used. Furthermore, MashQL does not support the utilization of resources of SWSs in the dataset integration.

As to the SWS, existing approaches introduced in the papers [10], [11], [12] mainly focus on designing models to automatically discover and compose SWSs. But these existing approaches do not allow users to integrate the resources of SWSs and normal RDF datasets.

To the best of our knowledge, only our ESVSW framework and its extension can support users to use the logical operators to exploratively search the Semantic Web resources, and to integrate the resources of the normal RDF datasets and the SWSs in the same work space.

III. THE ESVSW FRAMEWORK

The ESVSW framework provides a visual workspace (Fig. 1(1)) and two kinds of visual components, the class component (Fig. 1(4)~(8)) and the visualization component (Fig. 1(c), (e), (g)). Users can directly manipulate these two kinds of visual components to dynamically repeat the process of schema extraction, schema expansion, visual query definition, and query evaluation for obtaining the result visualization in an exploratory way until they find all the information they want to obtain.

The class component can automatically search and display two kinds of properties. The first kind property is called the resource property. Its value is a set of resources, which are identified by the URIs. The second kind property is called the attribute property whose value is a set of literals. In our ESVSW framework, if an instance of a class belongs to another different class, we call the latter class an overlapping class of the former class. The class component also can search and display the overlapping classes of its class. Each class component represents these two kinds of properties using two doughnut charts with different colors (Fig. 1(a), (b)). Each slice of a doughnut chart represents a property name. The angle of each slice is proportional to the amount of the subjects of its represented property. Users can switch the radio button on the class component to decide which kind of properties will be shown. In order to help users to read these charts easily, each class component provides a circle slider bar for restricting the amount of displayed properties. Each class component displays the overlapping classes by a drop-down list (Fig. 1(f)).

The visualization component provides a table and some types of charts. Users can directly connect such a visualization component to the class component for users to easily understand the retrieved instances of the class and the properties of these retrieved instances. For each visualization component, each visual object of a chart stores the URI of

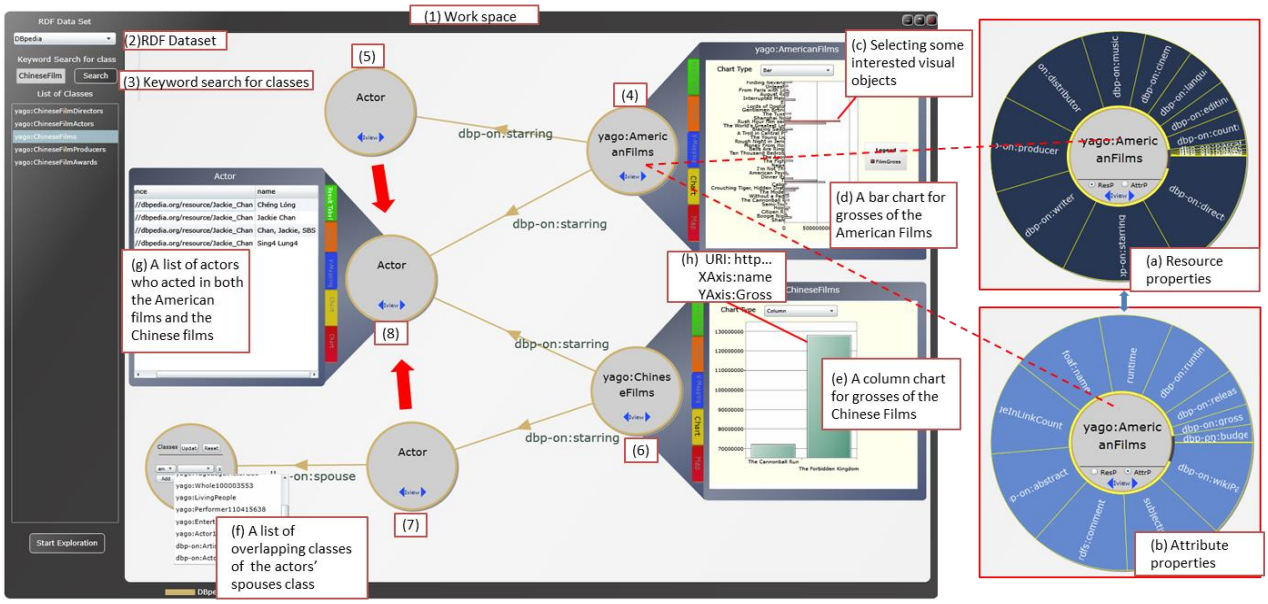

Figure 1. An outline of the ESVSW framework and its visual environment

the corresponding instance, and represents the value of the specified attribute of the corresponding instance (Fig. 1(h)).

Fig. 1 shows a simple example for illustrating the outline of the ESVSW framework. In this example, a director wants to make a new movie targeting both the Chinese market and the American market. In order to make both the Chinese people and American people have a sense of intimacy with this movie, the director need to perform following two tasks at least. The first task is to find out some actors who have acted in both the American films and the Chinese films. The second task is to search for the actors who have acted in either American films or Chinese films with the high grosses.

To perform the first task in the ESVSW framework, users can first retrieve two classes, the "American Films" (Fig. 1(4)) and the "Chinese Films" (Fig. 1(6)), thorough the keyword search (Fig. 1(3)) of a specified RDF dataset (Fig. 1(2)). Next, users can respectively extract their actor's classes (Fig. 1(5), (6)). Finally, users can directly combine these two "Actor" classes to define the required new "Actor" class (Fig. 1(8)) using the "AND" operator. Each actors of the newly obtained "Actor" class acted in both the American films and the Chinese films.

To perform the second task in the ESVSW framework, users can respectively retrieve the films names and their grosses in the "American Films" and the "Chinese Films". Next, the system will separately visualize the retrieved results using two different bar charts (Fig. 1(d), (e)). In each bar chart, users can directly select some long bars, which represent the files with high grosses, to obtain their relevant resources (Fig. 1(c)).

A. *Explorative Extraction and Expansion of the RDF Data Schema*

In our ESVSW framework, the RDF data schemata are generated either by using the sematic relationships among the instances of classes, or by combining different classes using the logical operators. The generated RDF data schemata are represented by graphs. Each node of the schema is a class component, and each edge of the schema is a semantic relationship between two classes. The ESVSW framework uses different colors to show the different schemata of the different RDF datasets.

Users can use some keywords to retrieve some classes, and select an interested class to begin the exploratory extraction and expansion of the local RDF data schemata. When the users click a class in the list of retrieved result (left side in Fig.1), the system will automatically load a new class component to represent the clicked class.

The ESVSW framework provides the following three operations for exploratory extraction and expansion of the local RDF data schema.

1) Defining a new class:

Each instance of this new class is a resource in the value of a user-selected resource property of some class in the expanded local RDF data schema. The system will add this newly defined class to the expanded local RDF data schema, and represent it using a new class component. As shown in Fig. 2, users can select and click the resource property "dbp-on:starring" (Fig. 2(3)) of the class "yago: AmericanFilms" (Fig. 2(1)) to obtain a new class "Actor" (Fig. 2(2)) whose instances are the resources in the value set of the resource property "dbpedia-on:starring".

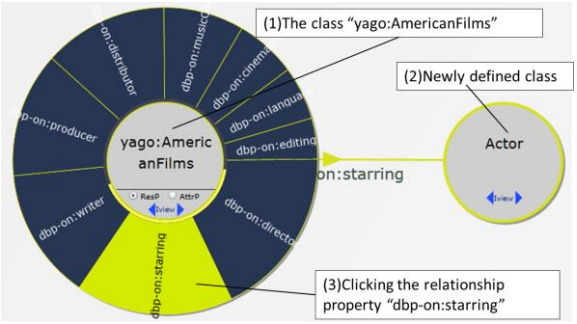

Figure 2. An example definition of a new class by using the value of a user-selected resource property

2) Defining a new class by combining two different classes of the expanded local RDF data schema using the logical operators:

The logical operators are AND, OR, and DIFFERENCE. Users can use one of these three operators to define a new class to respectively perform the intersection, the union, or the difference operation of two existing classes. As shown in Fig. 3, to perform this function, users can copy two class components (Fig. 3(1), (2)), and put them close to each other. Then the system will pop up a selection panel of logical operators (Fig. 3(3)). Users can select one logical operator to perform the required logical operation to combine these two classes. Then, the system will add this newly defined class to the expanded local RDF data schema, and represent it as a new class component (Fig. 3(4)).

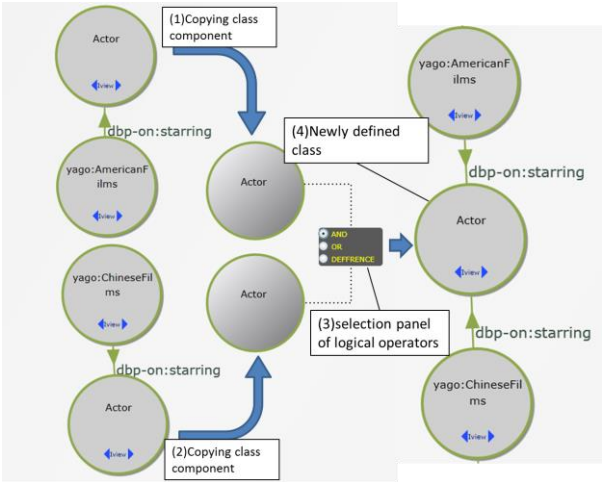

Figure 3. Logical operation on two classes

3) Modifying a class in the extracted RDF data schema using the result of a logical operation between itself and one of its overlapping classes:

Users can select the overlapping class from the drop-down list of the class component (Fig. 1(f)).

In our ESVSW framework, users can dynamically repeat the above three operations to exploratively extract and expand the local RDF data schemata.

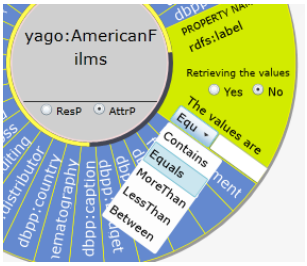

Figure 4. Visual definition of a query condition on a single class

B. *Visual Query Definition*

For each class component of the expanded RDF data schema, users can search its resource properties, its attribute properties, its overlapping classes, and its values of attribute properties. The queries for searching the resource properties, the attribute properties, and the overlapping classes for specific ones of some class are automatically generated by its class component, either when loading the class component or modifying the class on the class component.

To search for specific values of an attribute property, users can select and click an attribute property to invoke a query definition panel (Fig. 4). In this definition panel, users can select a button of "Yes" or "No" to specify whether to retrieve this property's values. Users also can select a predicate provided by each attribute property and input an argument to specify the value quantification condition. The provided predicates include "Contains", "Equals", "MoreThan", "LessThan", and "Between".

The ESVSW framework can automatically translate all quantification conditions into SPARQL queries and execute them. In each obtained SPARQL query, its "WHERE" clause consists of two parts. The first part represents the subset of the expanded RDF data schema. The second part represents the quantifications on the resource properties, the attribute properties, the overlapping classes, or the values of attribute properties.

```
SELECT    ?n3, ?n3_name
WHERE {
?n1  rdf:type  yago:AmericanFilms.
?n2  rdf:type  yago:ChineseFilms.
?n1  dbpprop:starring  ?n3.
?n2  dbpprop:starring  ?n3.
?n3  foaf:name  ?n3_name.
}
```

Corresponding to expanded schema

Corresponding to a restriction of an attribute property

Figure 5. A SPARQL query traslated from a visually defined query

```
SELECT    ?n3, ?n3_name
WHERE {
?n1  rdf:type  yago:AmericanFilms.
?n2  rdf:type  yago:ChineseFilms.
{?n1  dbpprop:starring  ?n3.}
UNION{?n2  dbpprop:starring  ?n3.}
?n3  foaf:name  ?n3_name.
}
```

(a)

```
SELECT    ?n3, ?n3_name
WHERE {
?n1  rdf:type  yago:AmericanFilms.
?n2  rdf:type  yago:ChineseFilms.
?n1  dbpprop:starring  ?n3.
?n2  dbpprop:starring  ?n.
FILTER (?n3!=?n)
?n3  foaf:name  ?n3_name. }
```

(b)

Figure 6. Two SPARQL queries converted form a SPARQL query shown in Fig.3.

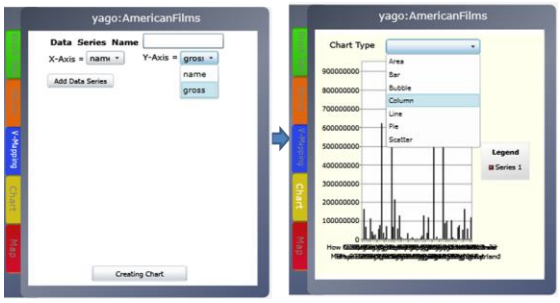

Figure 7. Definition of a visualization chart

For example, a user may want to search the actors of the "Actor" class (Fig. 1(8)) and their names as shown in Fig. 1. The SPARQL query for this example is shown in Fig. 5. In this example, "n1", "n2", and "n3" are respectively the ids of the "yago:AmericanFilms" class component (Fig.1 (4)) , the "yago:ChineseFilms" class component (Fig. 1(6)), and the "Actor" class component (Fig. 1(8)). From the first row to the second last row in the "WHERE" clause, the query describes the subset of the expanded schema consisting the three classes, the "yago:AmericanFilms" class (Fig.1 (4)) , the "yago:ChineseFilms" class (Fig. 1(6)), and the "Actor" class (Fig. 1(8)), and the relationships among them. The last row describes a query condition.

If we modify the logical operation of the two "Actor" classes (Fig.1 (5), (7)) in the example shown in Fig. 1 using either the "OR" operator or the "DFFERENCE" operator, the SPARQL query shown in Fig. 5 are respectively converted to the two queries shown in Fig. 6.

C. *The Visualization of the Searh Results*

In our ESVSW framework, the two kinds of visualization components are respectively shown in the three sheets, the "Result Table" sheet and the "Chat" sheet of each class component (Fig. 10). Each class component also provides the "V-Mapping" sheet to support users to directly map any retrieved attribute to the X_Axis or the Y_Axis of a chart. In the "V-Mapping" sheet, the names of retrieved attributes are listed in two drop-down lists (left side in Fig. 10). Users can select two attribute names from these two drop-down lists to perform the visual mapping.

For each visualization component, users can directly select some interested visual objects in the generated visualization, then the URIs stored in each selected visual object will be used as the query conditions to rewrite the queries.

D. *The Integration of Different RDF Datasets*

To integrate different RDF datasets in our ESVSW framework, Users can define a new connection between two different schemata by directly connecting two properties of two classes in the two schemata. As shown in Fig.8, we can extract the "yago:AmericanFilms" class from DBpedia and the "film" class from Linked movie Database. Then we can connect the property "foaf:name" of the "yago:AmericanFilms" class to the property "dcterms:title"

of the "film" class. The newly defined connection can transfer the values of the connected properties between the two classes. Through such a connection, users can search for the "costume_designer" of some film by clicking an instance of the class "yago:AmericanFilms" of the DBpedia. The "costume_designer" only exists in the Linked movie Database but not in the DBpedia.

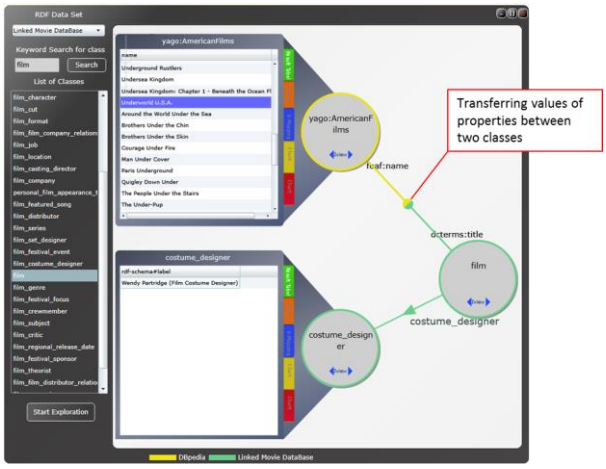

Figure 8. An example of defining a new integrated schema by combining two existing schemata

IV. THE EXTENDED ESVSW FRAMEWORK

In recent year, the amount of available SWSs increases rapidly. In order to effectively utilize Semantic Web resources, we need a generic platform for searching for and invoking SWSs, and integrating the resources of these SWSs with normal RDF datasets.

Since the SWSs repositories, such as iServe, provide SPARQL endpoints to enable users to query the SWSs using SPARQL queries, we can use our ESVSW framework to exploratively extract and expand the schema of the SWSs' descriptions in these repositories (e.g., the operation, the input, and the output of each SWS.). Using the expanded schema we can retrieve some interesting SWSs by directly specifying the descriptions of their inputs and/or outputs.

In an example shown in Fig. 10, we extract and expand the schema of SWS descriptions from iServe. The expanded schema is the graph shown in the upper part of Fig. 10. In the expanded schema, the property "modelReference" of the "Input" class and the "Output" class is used to annotate the input and output using existing RDF resources. Using this expanded schema, we can retrieve some services which can be used to search for music events by directly specifing the input as "geo:long" and "geo:lat" (Fig. 10 (1)), and specify the output as "MusicEvent" (Fig. 10 (2)). Each retrieved SWS is identified by a URI in the iServe (Fig. 10 (3)). In this example, only one SWS, called the "Geo.GetEvent", is found.

In order to invoke this retrieved SWS, and to integrate the resources of the normal RDF datasets with this SWS, we provide a new visual component called the service component in our extended ESVSW framework. This component enables users to quickly invoke the retrieved SWSs, and to connect this input and output to the expanded schemata of a normal RDF dataset.

When users Shift+click a retrieved SWS on the "Service" class component (Fig. 10(3)), the system will load a service component. The loaded service component will use the URI of the selected SWS to retrieve its interface using a SPARQL query (Fig. 9). In this SPARQL query the variable "RestURITemplate" corresponds to a template of the RESTful service request.

```
SELECT  ?input ?output ?RestURITemplate
WHERE{
< URI>          _:hasOperation      ?operation.
?operation      _:hasInput          ?in.
?operation      _:hasOutput         ?out.
?in             _:modelReference    ?input.
?out            _:modelReference    ?output.
?operation      _:hasAddress        ?RestURITemplate. }
```

Figure 9. A SPARQL query for retrieving interface of a SWS

The names of retrieved inputs and outputs are respectively displayed on the left side and the right side of the service component (Fig. 10(4)). Each input is a parameter of the retrieved RestURITemplate (Fig. 10(5)). User can specify the value of each input to define the Restful service request.

As the example shown in Fig. 10, we extract the "City" class (Fig. 10(6)) and the "MusicArtist" class (Fig. 10(7)) from DBpedia. Then we can respectively connect two inputs, the "geo:long" and the "geo:lat", of the retrieved service "Geo.GetEvent" to the two properties "geo:long" and "geo:lat" of the "City" class. Next, we connect the output "Artist" of the retrieved service to the property "foaf:name" of the "MusicArtist" class. After defining such connections, we can search for and select a city to retrieve all music events which will be held in this city, and the information about the artists who will attend the retrieved music events.

V. CONCLUSIONS

In this paper, we have proposed the ESVSW framework for Exploratory Search and Visualization of Semantic Web resources, and extend this framework to search and invoke SWSs. The ESVSW framework and its extension provide three kinds of visual components, the class component, the visualization component, and the service component. Users can directly manipulate the class component to dynamically extract and expand local schemata of the RDF dataset in an exploratory way starting from one or more classes of arbitrarily chosen Semantic Web resources. These schemata are generated either by using the sematic relationships among the instances of classes, or by logically combining different classes. Using these dynamically extracted schemata, users can visually specify queries and search the target Semantic Web resources for interesting information. Then users can also visualize the retrieved information by using the visualization component. They can dynamically

repeat this process of schema expansion, visual query definition, and query evaluation for obtaining the result visualization in an exploratory way until they find all the information they want to obtain. Users can also integrate the search results from different schemata by directly connecting class components of these schemata, or integrate the resources of SWSs and the resources of normal RDF datasets by directly connecting the service components to the class components in the schemata of the normal RDF datasets.

REFERENCES

[1] W3C, "Resource description framework," 2004. http://www.w3.org/TR/rdf-primer/.

[2] W3C, "Sparql query language for rdf," 2008. http://ww.w3.org/TR/rdf-sparql-query/.

[3] C. Pedrinaci and J. Domingue, "Toward the next wave of services: linked services for the web of Data," Journal of Universal Computer Science, http://oro.open.ac.uk/23073/

[4] C. Kiefer, A. Bernstein, and M. Stocker, "The Fundamentals of isparql: avirtual triple approach for similarity-based semantic web tasks," ISWC/ASWC, 2007, pp.295-309.

[5] I.O.Popov, M.M.C. Schraefel, G. Correndo, W.Hall, and N. Shadbolt, "Interacting with the web of data through a web of inter-connected lenses," LDOW, 2012.

[6] P. Heim and J. Ziegler, "Faceted visual exploration of semantic data," Human Aspects of Visualization, Lecture Notes in Computer Science, Springer, Berlin, Heidelderg, 2011, pp. 553-568...

[7] G. Kobilarov and I. Dickinson, "Humboldt: Exploraing linked data," LDOW, vol.369, CEUR Workshop Proceedings, 2008.

[8] D.A. Smith and N.R. Shadbolt, "Facetontology: Expressive descriptions of facets in the semantic web," JIST, 2012, pp.223-238.

[9] M. Jarrar and M.D. Dikaiakos, "Querying the data web: The mashql approach," IEEE Internet Computing, vol.14, no.3, 2010, pp. 58-67.

[10] S. Kona, A. Bansal, and G. Gupta, "Automatic composition of semantic web services", ICWS, 2007, pp.150-158.

[11] O. Hatzi, D. Vrakas, M. Nikolaidou, N. Bassiliades, D. Anagnostopoulos, and I.P. Vlahavas, "An integrated approach to automated semantic web service composition through planning," IEE T. Services Computing, 2012, pp.319-332.

[12] D.-H. Shin, K.-H. Lee, and T Suda, "Automated generation of composite web services based on functional semantics," J. Web Sem, 2009, pp.332-343.

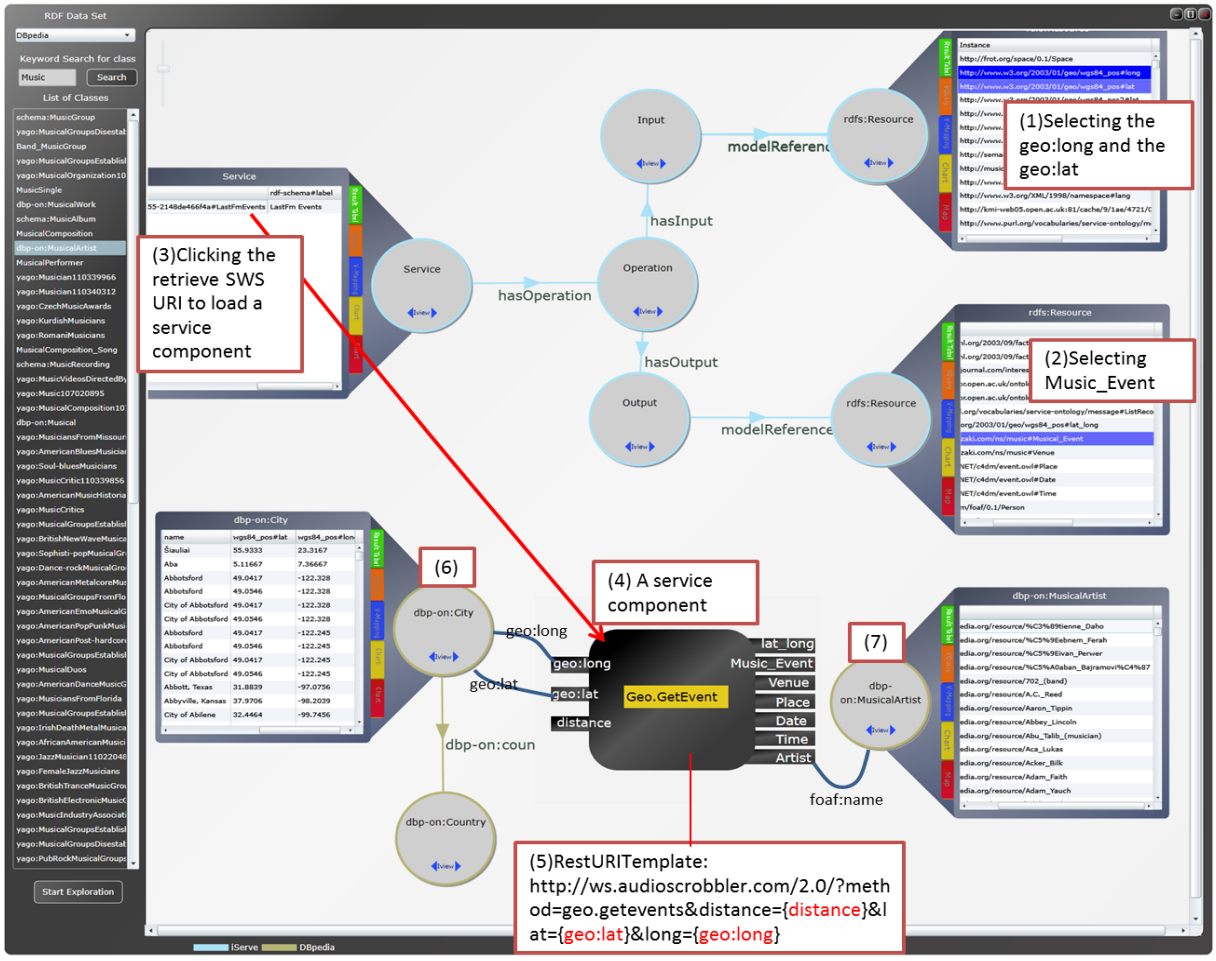

Figure 10. An example of defining a new integrated schema by combining two existing schemata