Brokered Approach to Federating Data using Semantic Web Techniques

Jeremy Siao Him Fa*, Geoff West†, David A. McMeekin‡ and Simon Moncrieff§
Cooperative Research Centre for Spatial Information
Department of Spatial Sciences, Curtin University, Bentley, Western Australia
Email: *jeremy.siao@live.com, †gawwwest@gmail.com ‡d.mcmeekin@curtin.edu.au §s.moncrieff@live.com

***Abstract*—There are many situations when spatial datasets that have different data structures, schema, and format need to be combined (e.g., to form a national road network from those supplied by different regions). Current methods dealing with combining independent data sources are manually intensive, and time inefficient suggesting the need for a more automated, efficient, and user centric approach. The main problem is due to syntactic, schematic, and semantic heterogeneities as the same domain data can be represented in different ways. Although tools solving syntactic heterogeneities are numerous, there is still a requirement to solve the underlying semantic problems. One way to alleviate semantic issues is by sharing a global schema for data providers to adhere to, but this method is unlikely to be implemented in multi-governmental federated countries such as Australia, as the data sets are owned by the different States and Territories. An automated brokered approach is proposed in this paper as a solution. This approach allows domain knowledge sharing across existing and future data providers, making it plausible where data sets are diverse and owned by different parties.**

Keywords–integration; semantic; wfs; federation; broker; owl-s.

I. INTRODUCTION

In this era of big data, data integration and interoperability has become an important technical challenge to research. The current methods dealing with interoperability issues have been manually intensive and costly [1]. Furthermore, the same datasets can be combined multiple times by different parties, leading to duplicated effort and conflation issues. The need for better and faster access to more user centric spatial data is further enhanced as spatial information is becoming more crucial to everyday life [2]. Such desires can be accomplished by a more automated way to combine data from different sources. This problem is most relevant for countries such as Australia, where the unification of the datasets has to be done at various governmental levels; Local Government Agencies (LGAs), States and Territories, and Commonwealth levels.

Schematic, syntactic, and semantic heterogeneities occur in datasets due to independent representation of the same domain data [3][4], and a lack of communication regarding the internal workings of organisations. An example of syntactic heterogeneities would be organisations storing data in different formats. As for schematic heterogeneity, a straight road can be stored as two points, or as a starting point and, a length and direction. As for an example of semantic heterogeneity, an 'aircraft' in organisation A might be a small light aircraft, while an 'aircraft' in organisation B might be an intermediate aircraft.

In order to solve these issues, and facilitate access to relevant information, interoperability at technical, syntactic, schematic, and semantic levels is required [5]. From all these, semantic interoperability is the underlying goal. Semantic heterogeneities occur because the same entity can have more than one representation or meaning [5][6][7]. Dealing with semantic problems requires domain knowledge sharing and common vocabularies. Technical interoperability, on the other hand, is already widely used; Hypertext Transfer Protocol (HTTP) used on the Web allows multiple machines to communicate via the same protocol. As for syntactic interoperability, the use of same data formats (e.g., JavaScript Object Notation (JSON), Extensible Markup Language (XML), shape files [8], Resource Description Framework (RDF)) or seamlessly transforming one data format to another are possible solutions. Tools are already available for the latter task, such as Feature Manipulation Engine (FME) [9].

Schematic or structural heterogeneities happen when there is a difference in the data schema or structure [10][7]. This is due to the datasets being developed independently, and thus using different and varying structures for the same or similar concepts [3]. Although the main consensus to solve schematic heterogeneities is through standard schemas, it has been shown that they are limited in success unless there are strong incentives to use such standards [3]. Such an approach can be observed in the Infrastructure for Spatial Information in the European Community (INSPIRE) initiative in Europe [11]. This initiative requires the different data providers to change their existing business model to adhere to a global schema, as well as added legislations and policies to ensure data quality. Data providers are further required to translate their data (new and old) to the new format. The restrictions imposed could also potentially lead to loss of information.

As such, a unifying process not requiring much change from the data providers, and allowing them to keep their existing models would be preferable. Other possible approaches include (1) point to point, (2) centralised, (3) aggregated, and (4) brokered [5]. The point to point approach requires the user to do all the unification without any intermediary between them and the data providers. This approach is very time consuming, manually intensive, and leads to duplicated effort and conflation issues. The second approach happens where all the data is stored, handled, and provided by a single organisation, which is improbable to happen in federated countries. The aggregated approach requires a data warehouse to aggregate and store the datasets from the multiple data sources. The cost of such a method scales as more data sources are included, alongside the increased duplication of data. The fourth approach is explored in this paper and will be discussed in the next few paragraphs.

In Australia, where the datasets are owned by different private agencies, and States and Territories, data aggregation is the method currently used to share spatial data across the different jurisdictions. This approach though does not provide

 ISBN: 978-1-61208-469-5

the most up to date information, being out of date anywhere from three months to six months [12]. As such, to provide the most up to date information, while not restricting data providers, a broker approach is explored in this research.

A brokered approach makes use of a centralised mediator to transform the data from multiple sources onto an agreed schema [5]. Its aim is to deal with semantic heterogeneities via translations of diverse conceptualisations [13]. Most of the effort is put on the mediator, while the providers do not have to change their existing business model as long as they publish their schema and when they are updated. As no change in business models is required, this pattern caters for both current and future data providers. An Example of such an approach can be found in EuroGEOSS [14] in Europe.

In this paper, we propose to use an automated brokered approach to seamlessly unify different datasets from different governmental levels. Using semantic web technologies, it discovers data suppliers and adapts to their schemas dynamically. Furthermore, it does not impose change in existing business models, and gives the most up to date unified information from multiple sources. The main outcome is to enable an easy way to access specific distributed spatial information, and alleviate users from manually translating datasets.

This paper first presents relevant background material in Section 2. Related work is presented in Section 3, followed by a description of the proposed automated brokering system in Section 4. Section 5 details the current state of the project, and is followed by conclusions and plans for future work in Section 6.

II. Background Information

A. Spatial Web Services

Web Feature Services (WFS) are popular to provide access to spatial features. WFS is an Open Geospatial Consortium (OGC) web standard interface to provide remote querying to a collection of geographic features [15]. Although WFS provide data interoperability by offering different data output format, it is not designed to support schematic interoperability [16].

A WFS uses a multi layered approach to allow for querying of its capabilities (GetCapabilities), the schema of a particular feature (DescribeFeatureType), and the actual data set of a feature (GetFeature).

B. Ontologies

Dealing with semantic heterogeneities with an automated brokered approach requires ontologies to represent knowledge so as to understand how to interpret semantic differences and enable transformations. An ontology is defined as a concept where domains of interest share their understanding with each other [17][18][19]. In more technological usage, ontologies are utilised for sharing, representing, and storing knowledge in the form of data [20]. Essentially, an ontology is a graph that can be traversed in order to find links between concepts of a given domain's knowledge. Ontologies can be serialised in multiple text formats such as XML, Turtle (ttl), and Notation 3 (n3). Any of these formats can be easily transformed to another, making them syntactically interoperable.

A common semantic web language for ontologies is that of the Web Ontology Language (OWL) [21]. OWL is a World Wide Web Consortium (W3C) standard whose main purpose is to provide formalisms and to allow for knowledge representation. In this research, we make use of OWL-S [22], which is based on OWL to describe web services.

OWL-S was primarily designed to describe any potential web services, and thus was adapted to cater for WFS specifically. The modification follows the works of Stock et al. [23] closely but as only a partial of their modified ontology was available, many changes had to be extrapolated from their work. The OWL-S ontology therefore, is an ontology that can be adapted to describe WFS.

The OWL-S ontology is made up of three main components [22]:

1) The service profile used to identify, advertise, and discover web services. This is where a program can find out if a particular web service is what is needed;
2) The process model explains how the service works, the input, output, and processes for its different functions. This is where a program finds out how to use a particular service, what to input, and what is received back after the process; and
3) The grounding provides details on how to interact with the web service via messages.

The OWL-S ontology gives a strong foundation to describing web services due to it being a standard, allowing for future planning, and for much shareability.

III. Related Work

Work related to this research include various adaptations of the OWL-S ontology, novel ways to approach the brokering method, proposals to facilitate sharing of data, and attempts at solving ambiguities regarding heterogeneous datasets.

An adaptation of OWL-S includes a cloud service broker [24]. The authors define a cloud broker as 'an entity that manages the use, performance, and delivery of cloud services, and negotiates relationships between cloud providers and cloud consumers'. Their challenge is the heterogeneous nature of multiple service providers. Cloud service specifications are not standardised and promote semantic heterogeneity. By using OWL-S, the cloud service broker is able to dynamically discover complicated services whose attributes are constrained. The constraints of the services are represented using the Semantic Web Rule Language (SWRL). The cloud service broker is able to solve varying tasks of different levels through multiple case studies but requires the different service providers to use a shared ontology. Ngan and Kanagasabai [24] state that ontology alignment and learning (part of this research) can address that issue.

Another adaptation related to this paper involves implementing the OWL-S ontology to describe WFS and Web Map Service (WMS) [23]. To cater for Open Geospatial Consortium (OGC) compliant web services, the OWL-S WSDL grounding was changed to a simplified OGC equivalent grounding. Their work shows detailed analysis of WFS and WMS, and an implementation of their OWL-S adaptation demonstrated its practicality in the marine domain. Although they state that OWL-S is a 'fairly cumbersome specification', their work showed that it can be adapted to OGC compliant web services.

Yue et al. [25] explores automated geospatial web services' composition using semantics. In their approach, the authors

 ISBN: 978-1-61208-469-5

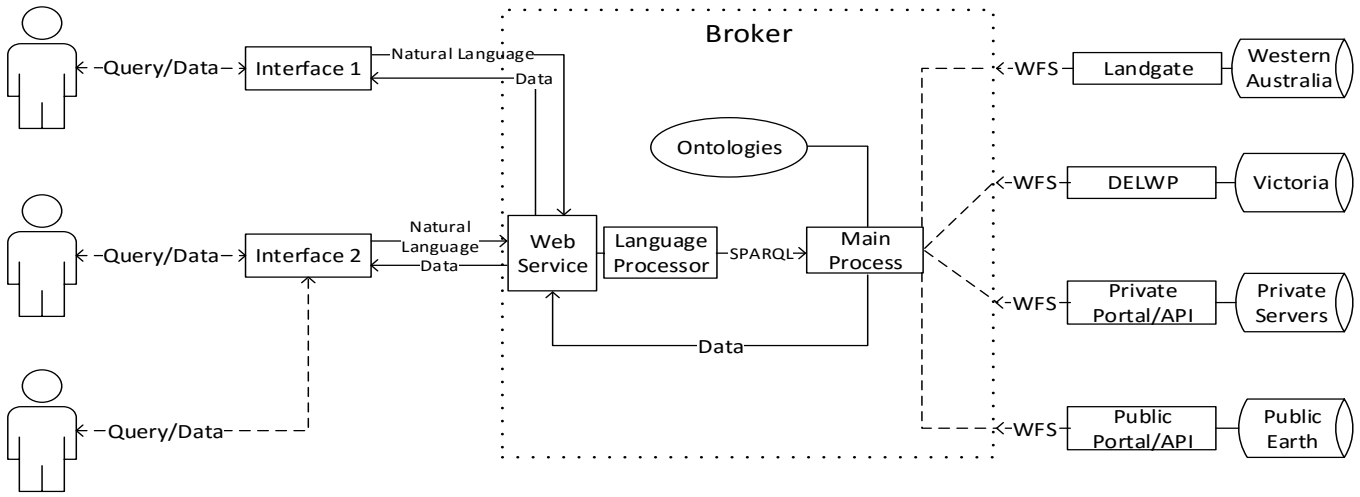

Figure 1. Automated Broker System.

designed and used 'DataType', 'ServiceType', and 'Association' ontologies as semantic schemas. Although some results have been obtained in their use case, the automation of the approach needs improvement; domain experts are required in some aspects and the ontologies' reasoning needs more work.

At a lower level, work has been done aiming at matching WSDL descriptions to that of OWL-S semantic annotations [26]. Facilitating the move from WSDL description of web services to OWL-S would promote web services discovery. Their method includes an ontology repository where a matching algorithm finds the best match related to specific concepts in a WSDL description. Using various designed and real examples, their algorithm has been validated to be practical and to reduce time and effort.

A different approach is using ontologies to provide a conceptual overview of the data sources. This particular approach is called Ontology-Based Data Access (OBDA). Its aim is to provide a query-able ontological view of the data sources to the users. Some of the current systems harnessing OBDA includes the Optique System [27] and MASTRO [28]. A recent evaluation the OBDA approach found that it can be 'orders of magnitude faster than standard triple stores' [29], using proper optimisation techniques. However, OBDA can also perform poorly if proper techniques are not used.

A real life application of OBDA using the MASTRO system was explored by Antonioli et al. [30]. They found that the mapping definition from the Italian Department of Treasury case study had to essentially be done manually. They had to manually analyze the structure of the data sources to understand the data semantics. This though, only had to be done once using the MASTRO system, and once for any new sources [30]. Some query rewriting optimisation has also been suggested, making previously impractical methods possible.

Regarding ambiguities and uncertainties, Yang et al. [31] studied the usage of Bayesian Networks. According to them, Bayesian theory provides a principled representation of uncertainty, alongside logic to unify observations with previous knowledge, and learning theory for refining ontologies. Although their approach had improved efficiency and accuracy in regards to geospatial web services, it targeted mainly discovery of such services. Furthermore, manual work in conjunction with venture capitalists were required to build the raw causal map needed in their work.

IV. Automated Broker for Unifying Uncontrollable Heterogeneous Data Sources

To facilitate access to multiple data sources, it is required to have (1) web services, (2) descriptions of the web services, and (3) a federated model.

For the web services, two WFS (described using OWL-S) are used - Landgate and Department of Environment, Land, Water and Planning (DELWP), Victoria.

As for a federated model, the Foundation Spatial Data Framework (FSDF) is used in this paper. The FSDF is a national level dataset developed by the Australian and New Zealand Land Information Council (ANZLIC) [5]. Its aim is to provide a number of foundational data themes (geocoded addressing, administrative boundaries, positioning, place names, land parcel and property, imagery, transport, water, elevation and depth, and land cover), each one consisting of a number of datasets [32]. A part of the administrative boundary dataset was used as a use case study.

Datasets in the FSDF have been modelled using the Unified Modelling Language. The FSDF was transformed to an OWL equivalent using the tool Protégé [33] to allow reasoning and querying using SPARQL Protocol and RDF Query Language (SPARQL) - an RDF query language.

As we are using semantic web concepts, ontologies will thus be the main technique used to describe the knowledge needed to link up the web services' descriptions, and federated model. This combination will be referred to as ontology Θ from here on.

A. Broker Architecture

The structure of the proposed brokered system has the federated model acting as a unified view for the user, the broker in the middle, and the web services at the end. Multiple users can use the broker system through a web service (e.g., WFS, Web Processing Service) and adapt it to their own needs. The exposed web service then translates the user query into a

 ISBN: 978-1-61208-469-5

SPARQL equivalent which is then used to query the ontologies and relevant data sources.

This structure is portrayed in Figure 1 where the left hand side are multiple users using the same or different interfaces plugged into an exposed web service, the middle is the broker with its various components, and the right hand side are the data providers and their web services. The broker system is the mediator between the users and the web services, and while the users can interact freely with the broker they do not need knowledge of the data sources.

In general, the broker has to deal with:

1) Querying relevant web services;
2) Processing the user's query; and
3) Combining the differing data sets from the relevant suppliers.

B. Querying Web Services

In this paper, web services used are WFS that adhere to standards (i.e., OGC), and have available URLs. The capabilities of the WFS can be analysed to identify if the serviced features are related to the query posed. For example, parsing the 'Title' and 'Description' tags, a matching algorithm can determine if a particular feature is relevant or not. If it matches, then the WFS capabilities are modelled and stored in the OWL-S ontology for future use. This can be observed in Figure 2 where the OWL-S ontology is connected to a GetCapabilities ontology.

After obtaining the capabilities of the service, and asserting that some typeNames (name given to a specific feature) are useful to the user, a DescribeFeatureType call is made to the service. This call returns a schema describing the meta-data stored by the web service. The schema is then parsed for meta-related to the user's query (e.g., to see if a particular attribute 'name' is available in a particular feature). The DescribeFeatureType schema of that particular feature is then modelled and stored in the GetCapabilities node found in Figure 2.

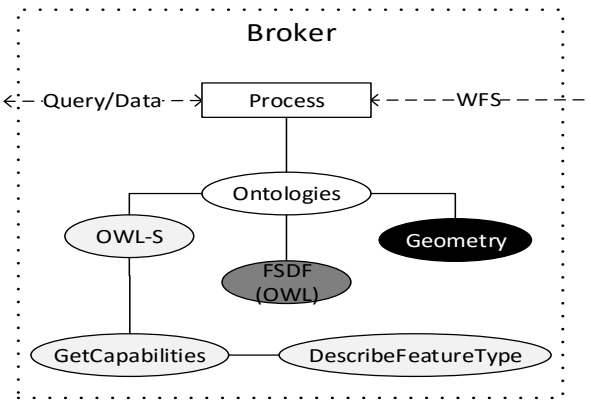

Figure 2. Ontologies Used.

To automate the process of linking the DescribeFeature-Type schema to the broker's ontology, Extensible Stylesheet Language Transformations (XSLT) are used. These are programmable style sheets that allow transformation of XML to any other format. Doing so enable an automatic transformation of the DescribeFeatureType XML to RDF providing an on the fly linkage to the ontology used in the broker.

Figure 3 shows the sequence of actions needed to query a number of URLs.

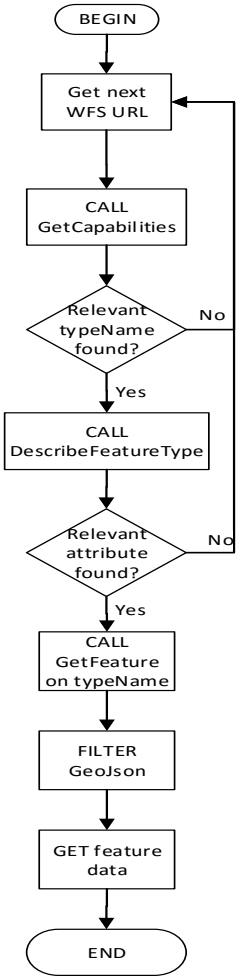

Figure 3. Overview Flowchart.

The resulting RDF triples are added to ontology Θ. Each WFS URL now has a feature, and that feature has its meta-data linked to it (Refer to 'slip:LGATE-069_Type' in Figure 6).

Once the required attribute has been obtained, the link is then added to the ontology as an equivalent term. For example, the term *StateElectoralDivision* from the FSDF, has an individual from Landgate's LGATE-069 feature type, and the attribute *fsdf_name* has an equivalent link to LGATE-069's name attribute in the ontology.

Figure 6 demonstrates the ontology developed. The light grey parts are from the OWL-S ontology, its extension can be seen from the white ellipses and rectangles, the black part is the start of an OWL implementation of the ISO 19107 Spatial Schema [34], and the dark grey part is the ontology adapted from the FSDF model.

To link up all three ontologies - pseudo-FSDF, OWL-S, and geometry, the following changes have been made:

 ISBN: 978-1-61208-469-5

1) Adding a WFSAtomicProcess class as a subclass of the OWL-S AtomicProcess;
2) Adding process name individuals for the WFSAtomicProcess (i.e., GetCapabilities, DescribeFeatureType, and GetFeature);
3) Linking the DescribeFeatureType and GetFeature WFSAtomicProcesses to the individual 'slip:LGATE-069' as input;
4) Adding WFSProfile as a subclass of OWL-S' ServiceProfile;
5) Adding DescribeFeatureType's meta-data as attributes for slip:LGATE-069_Type using the 'hasFeatureTypeComponent' object property; and
6) Using the geometry ontology for the 'bbox' meta-data for slip:LGATE-069_Type.

C. Processing the Query

A user's natural language query is transformed to a SPARQL equivalent query. SPARQL allows constraints to be placed upon the queries, enabling more detailed querying.

In order to process a SPARQL query, it is first required to extract its components. In this paper, the two main components will be the `SELECT` and `FILTER` clauses. The `SELECT` clause is assumed to mean a specific feature type, and the `FILTER` clause is assumed to mean a specific instance of the feature type found in the `SELECT` clause. By making this differentiation, there are thus two main types of queries: (1) a generic query (2) a detailed query:

1) The generic query is associated with the `SELECT` clause, and is assumed to refer to a specific feature type, not an instance. As such, a GetCapabilities call to the WFS is all that is required. From the retrieved XML, it is then possible to filter the title, description, and keywords to match the query's details; and
2) The detailed query is determined when a `FILTER` clause is found. The clause will determine which details the user is looking up in a particular feature type. The generic query needs to be processed first though. That is, only after getting the feature type associated with the `SELECT` clause, can a particular instance from that feature type be found.

From both generic queries and detailed queries, the base ontology Θ can then be expanded as the WFS is being explored.

1) Matching Generic Queries: A generic query is determined by the `SELECT` clause of a SPARQL query. That clause is assumed to be direcly linked to a feature type in a WFS. An example of such a query is depicted in Figure 4.

```
SELECT ?feature
   WHERE {
         ?feature rdf:type ex:StateElectoralDivision .
   }
```

Figure 4. SPARQL for Matching Generic Queries.

The query is assumed to be looking for a feature type that's related to 'StateElectoralDivision'. For such queries, the GetCapabilities of each WFS, has to be checked. The features' 'name', 'title', 'abstract', and 'keyword' go through a set of matching algorithms in order to find any similarity with the feature 'StateElectoralDivision' from this example. These particular fields have been chosen as the 'name' is a mandatory field in any feature, the 'title' is intended to 'briefly identify the feature type', the 'abstract' provides more descriptions about the feature type, while the 'keyword' is intended to aid catalog searching [15].

Figure 5 demonstrates the sequence of processes that take place, after retrieving the URL of the WFS from the ontology. After the URL is retrieved, a GetCapabilities call is made to the server. The returned XML is parsed for the fields described previously, and a matching algorithm is run to find any similarities. Given that a particular feature is found to be similar, the next step is to make a DescribeFeatureType call for the matching feature, giving back a schema (e.g., XML Schema Definition). The schema is parsed for any attributes, and these are created in the broker's ontology as 'hasFeatureTypeComponent' links to the feature (Refer to Figure 6).

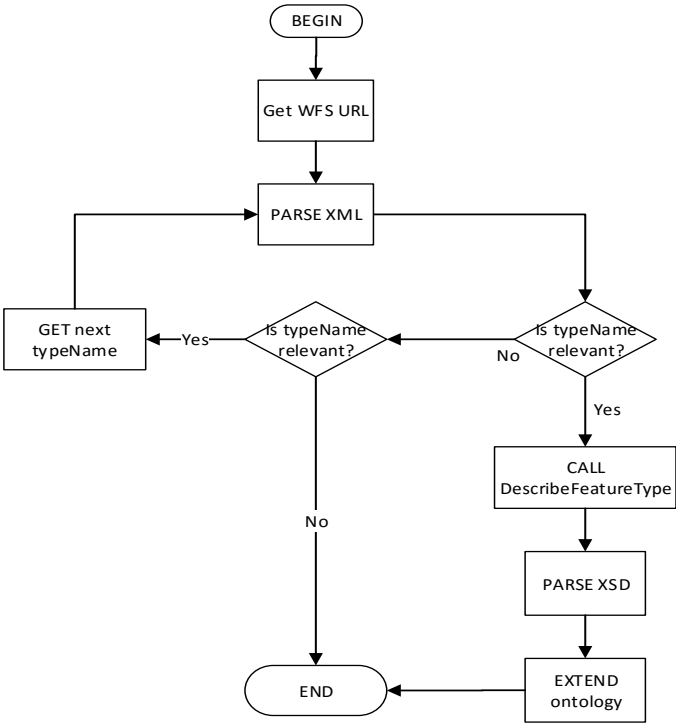

Figure 5. Generic Queries Matching Flowchart.

2) Matching Filter Queries: A filter query determined by the `FILTER` clause of a SPARQL query. That clause is assumed to be directly linked to a specific attribute in a WFS' feature. An example of such a query is depicted in Figure 7.

```
SELECT ?feature
   WHERE {
         ?feature rdf:type ex:StateElectoralDivision .
         ?feature ex:name ?name .
         FILTER( name = 'Albany' )
   }
```

Figure 7. SPARQL for Matching Filter Queries.

It is assumed the query is looking for a feature type whose attribute relating to 'name' is related to 'Albany'. After parsing the generic query of finding a typeName similar to

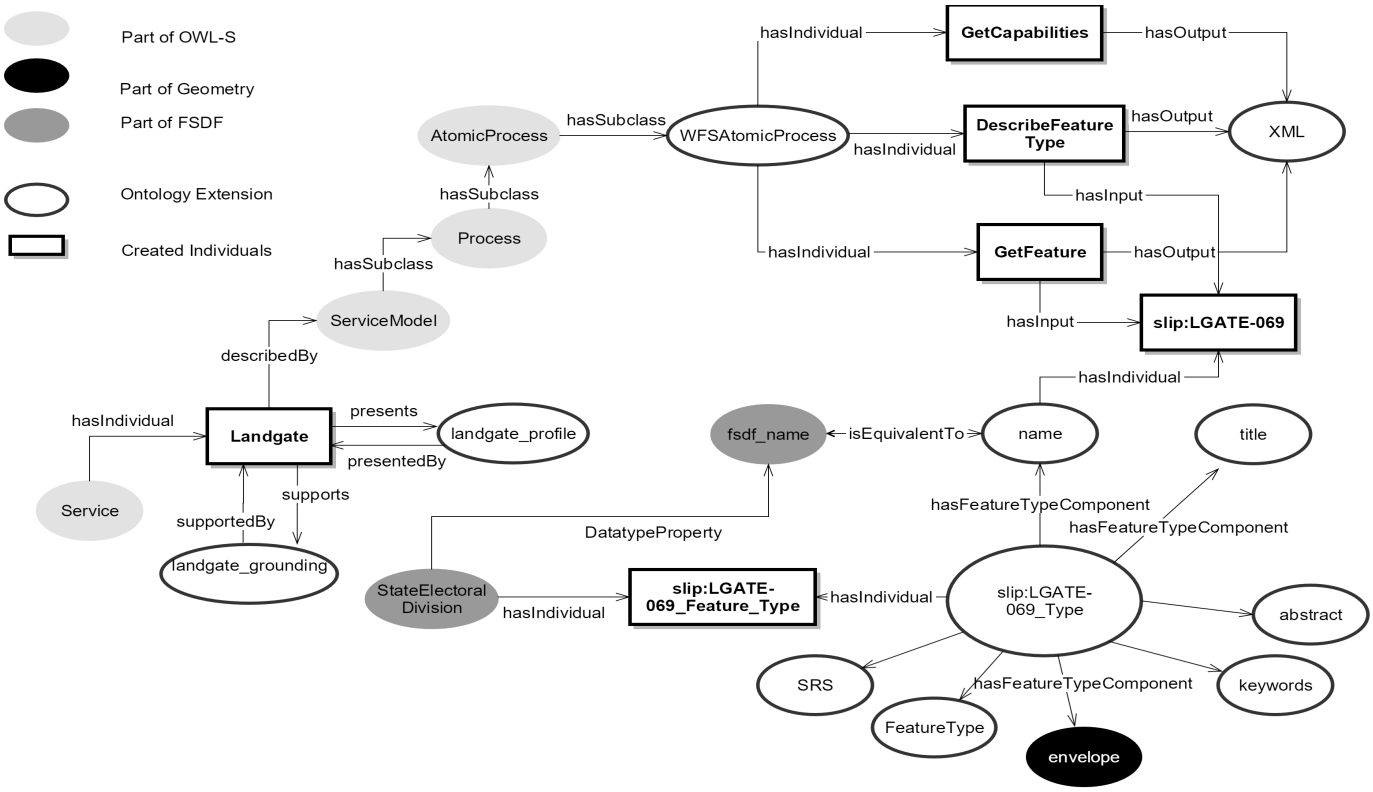

Figure 6. Adapted Ontology (Θ) Dependence.

TABLE I. COMPARISON OPERATORS.

WFS Comparison Operator	SPARQL Comparison Operator
PropertyIsEqualTo	=
PropertyIsNotEqualTo	!=
PropertyIsLessThan	<
PropertyIsGreaterThan	>
PropertyIsLessThanOrEqualTo	<=
PropertyIsGreaterThanOrEqualTo	>=
PropertyIsLike	N/A
PropertyIsBetween	< && >

'StateElectoralDivision', the next steps are demonstrated in Figure 8. All the meta-data are retrieved from the particular feature type using a DescribeFeatureType call. The schema obtained is filtered to find a similar attribute to that of 'name'. Given that a similar attribute is found, the next step is to call GetFeature from the web service, and filter the attributes for 'Albany'.

3) SPARQL Filters to WFS Filters: Instead of filtering a whole feature type within the broker, filter parameters can be passed into a GetFeature call to the WFS. It enables the provider's WFS to run a filter search server side, removing the filtering overheads from the broker. For simplicity, only comparison operators (e.g., =, !=, <, >, etc.) are discussed in this paper but others such as 'Union', and regular expressions are possible.

Table I shows the comparison operators that are common in WFS and SPARQL.

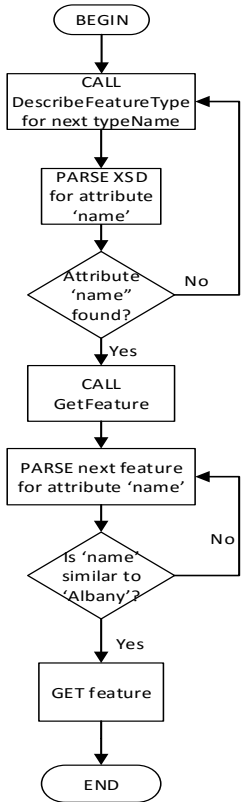

Figure 8. Filter Queries Matching Flowchart.

Considering Table I, the SPARQL query can thus be converted to a WFS call equivalent.

```
SELECT ?feature
    WHERE {
            ?feature rdf:type ex:StateElectoralDivision .
            ?feature ex:fsdf_name ?name .
            FILTER( name = 'Albany' )
    }
```

Figure 9. SPARQL Name matching.

Given that 'LGATE-069' is the equivalent to 'StateElectoralDivision', and that 'name' is the equivalent to 'fsdf_name', then the WFS call would be as depicted in Figure 10.

```
www2.landgate.wa.gov.au\ows\wfspublic_4283\wfs?SERVICE=WFS&
VERSION=1.0.0&REQUEST=getFeature&typeName=LGATE-069&Filter=

<Filter><PropertyIsEqualTo>
        <PropertyName>name</PropertyName>
        <Literal>Albany</Literal>
</PropertyIsEqualTo></Filter>
```

Figure 10. Name Filtering Call.

Using the 'PropertyIsEqualTo' though, is restrictive, as the precise name of the anything stored has to be known. This issue can be solved by using the 'PropertyIsLike' operator. The same query can then be rewritten as depicted in Figure 11.

```
www2.landgate.wa.gov.a\ows\wfspublic_4283\wfs?SERVICE=WFS&
VERSION=1.0.0&REQUEST=getFeature&typeName=LGATE-069&Filter=

<Filter><PropertyIsLike wildCard='*' singleChar='!'>
        <PropertyName>name</PropertyName>
        <Literal>*Albany*</Literal>
</PropertyIsLike></Filter>
```

Figure 11. PropertyIsLike Call.

That call would look for any name attribute that has 'Albany' in it.

For multiple word names such as 'Alfred Cove', a simple algorithm permutating all the ways it can be written can be used. For example, the identified ways 'Alfred Cove' can be written as are:

nolistsep

1) Upper case (ALFRED COVE);
2) Lower case (alfred cove);
3) Upper case with underscore (ALFRED_COVE);
4) Lower case with underscore (alfred_cove);
5) Camel case with space (Alfred Cove);
6) Camel case without space (AlfredCove); and
7) Camel case with underscore (Alfred_Cove).

All the seven ways of naming conventions can be programmed to fit in the WFS GetFeature filter URL as depicted in Figure 12.

D. Converting XSD to RDF

Converting XML Schema (XSD) to RDF format automatically has been achieved in various disciplines [35][36][37].

In this research, XSLT was used as it enables the transformation of an XML Schema to RDF without needing any human assistance. The XSLT was developed and used successfully but does contain some limitations, such as data type duplicates. It was utilised to automatically link the DescribeFeatureType schema of a WFS to the broker's ontology as depicted in Figure 2 and Figure 5. The generic steps undertaken in the XSLT are:

```
www2.landgate.wa.gov.au\ows\wfspublic_4283\wfs?SERVICE=WFS&
VERSION=1.0.0&REQUEST=getFeature&typeName=LGATE-069&Filter=

<Filter><Or>
        <PropertyIsLike wildCard='*' singleChar='!'>
                <PropertyName>name</PropertyName>
                <Literal>*ALFRED COVE*</Literal>
        </PropertyIsLike>
        <PropertyIsLike wildCard='*' singleChar='!'>
                <PropertyName>name</PropertyName>
                <Literal>*alfred cove*</Literal>
        </PropertyIsLike>
        <PropertyIsLike wildCard='*' singleChar='!'>
                <PropertyName>name</PropertyName>
                <Literal>*ALFRED_COVE*</Literal>
        </PropertyIsLike>
        <PropertyIsLike wildCard='*' singleChar='!'>
                <PropertyName>name</PropertyName>
                <Literal>*alfred_cove*</Literal>
        </PropertyIsLike>
        <PropertyIsLike wildCard='*' singleChar='!'>
                <PropertyName>name</PropertyName>
                <Literal>*Alfred Cove*</Literal>
        </PropertyIsLike>
        <PropertyIsLike wildCard='*' singleChar='!'>
                <PropertyName>name</PropertyName>
                <Literal>*AlfredCove*</Literal>
        </PropertyIsLike>
        <PropertyIsLike wildCard='*' singleChar='!'>
                <PropertyName>name</PropertyName>
                <Literal>*Alfred_Cove*</Literal>
        </PropertyIsLike>
</Or></Filter>
```

Figure 12. ProperIsLike Name Filtering Call.

1) For each XSD targetNamespace, convert them to RDF namespaces;
2) For each XSD complexType, convert them to OWL Class;
3) For each XSD extension within a xsd complexType, convert the class to a subclass of the extension;
4) For each sequence within a XSD complexType, convert them to an OWL subClassOf owl:Restriction;
5) For each XSD element within a XSD sequence, convert them to an owl:onProperty with the rdf:resource being the base URL plus the element's name;
6) For each minOccurs within the xsd:element, convert them to owl:minCardinality;
7) For each maxOccurs within the xsd:element, convert them to owl:maxCardinality; and
8) For each type within the xsd:element, convert them to an owl:Datatype with the rdfs:range being the xsd:type and the rdfs:domain being the owl:Class;

Following these mapping guidelines, an RDF file was automatically generated to extend ontology Θ.

E. Combining Differing Data Sets

The combination of varying data sets is done on a per-query basis. For each of the queries tasked by the user, the broker goes through the two steps above: (1) querying web

services, and (2) processing the user query if relevant terms in the query are not already existent in the ontology. For example, if the general term in the query's `SELECT` is found to have an equivalence of 'LGATE-069' from Landgate in ontology Θ , then further processing is not required.

As these processes are carried out, the ontology would be growing to have further links and equivalent terms, rendering the processing less and less required as the results of previous queries are stored. That semantic aspect makes the broker highly scalable, as the system constructs more links as more queries are processed.

F. Error Handling

Given that an error occurs in calling a WFS service, an error message would be returned either in JSON or Hypertext Markup Language (HTML). A JSON error message can be parsed to find the specific cause of the problem (e.g., feature does not exist), while the error code of the HTML provides a direct indication of it (e.g., Error 500 for internal error). Depending on the cause of the problem, two main scenarios can happen (1) the ontology can be modified to cater for the changes, and (2) the error cannot be handled properly.

Case number one happens when the schema of the WFS is changed and thus the ontology is not up to date with the changes. This can be resolved by calling the WFS GetCapabilities, running a matching algorithm to the broker's ontology and compare for discrepancies. The mismatches can then be resolved by updating the ontology to reflect the new schema.

Case number two happens when there is either a problem on the provider's side (e.g., server is down), or a problem on the user's side (e.g., user input error). In both cases, nothing can be done as these are errors of the broker's scope, and the only solution is to notify the user about it.

G. Performance

Using the on-the-fly approach ensures that only a minimal amount of information is stored on the broker's side. Most of the main data (e.g., coordinates, shapes, etc.) remain at the data source. The only stored information are the ontologies that link data providers' gateway to the global ontology. With the example of WFS, only information up to the secondary level (DescribeFeatureType) is stored as part of the ontology, the third level, which contains most of the data, is left at the source. The ontologies can moreover be distributed on the cloud; various servers can be used, making storage problems minimal. Furthermore, caching will be used to speed searching, and parallel computing or torrenting technologies could be explored as well.

V. Results

A case study of the broker system was implemented using the Python programming language [38]. Python has various libraries and frameworks already available for usage. For example, the library rdflib allows the usage of RDF and ontology graphs, to query, create, edit, and import. Reusability of such libraries is a forte of Python, especially for proofs of concept. Furthermore, the framework Django [39] has been used as it enables easy set up of a web interface to implement a server locally.

Figure 13. Brokered System Result.

A. Current State

The broker system implemented provides a visual display for the OWL implementation of the FSDF. A list of classes - with their respective attributes from the FSDF UML - is shown. Once a class and an attribute are selected, the user can specify a value to query. A start button starts the querying process, and a map is updated automatically when any result is obtained.

Figure 13 shows the returned result from two different States (Western Australia and Victoria). The red marks are the locations' boundaries, which can be seen clearer in Figure 14 and Figure 15 respectively.

VI. Conclusion and Future Work

This paper discussed the development of a broker system to automatically and virtually consolidate various spatial data sources on a per-query basis for easier user consumption. The aim of the broker is to act as a mediator between users and data sources, to combine various heterogeneous data sources.

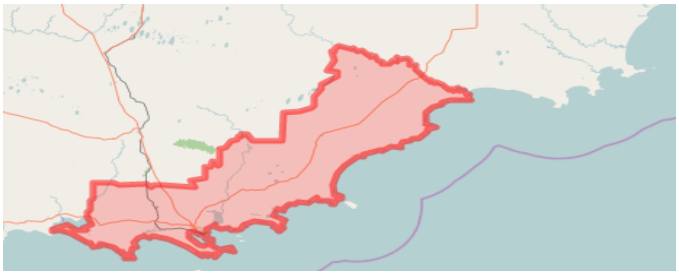

Figure 14. Albany (WA) Result.

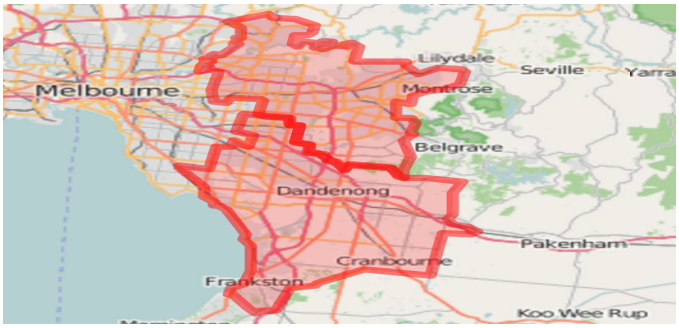

Figure 15. Eastern Metropolitan Area (VIC) Result.

This work is far from being completed and it is planned to extend the broker system to more agencies and operations. Reasoning to cope with ambiguities is to be developed, as well as, the automatic expansion of the ontologies. Other future work will include more filtering abilities such as intersection of regions, and queries regarding different data sets such as housing and forests. Furthermore, the web services to be

 ISBN: 978-1-61208-469-5

implemented have to be diversified, as well as the themes from the FSDF.

ACKNOWLEDGEMENT

This work has been supported by the Cooperative Research Centre for Spatial Information, whose activities are funded by the Business Cooperative Research Centres Programme.

REFERENCES

[1] C. Terblanche and P. Wongthongtham, "Ontology-based employer demand management," *Software: Practice and Experience*, pp. 1–46, Apr 2015.

[2] ACIL Tasman, "The Value of Spatial Information," ACIL Tasman Pty Ltd, Tech. Rep. March, 2008.

[3] A. Halevy, "Why Your Data Won't Mix: Semantic Heterogeneity," *Queue*, vol. 3, no. 8, pp. 50–58, oct 2005.

[4] D. J. Abel, B. C. Ooi, K.-L. Tan, and S. H. Tan, "Towards integrated geographical information processing," *International Journal of Geographical Information Science*, vol. 12, no. 4, pp. 353–371, jun 1998.

[5] P. Box, B. Simons, S. Cox, and S. Maguire, "A Data Specification Framework for the Foundation Spatial Data Framework," CSIRO, Australia, Tech. Rep., 2015.

[6] J. X. He, "An Ontology-Based Methodology for Geospatial Data Integration," Ph.D. dissertation, uOttawa, 2010.

[7] I. F. Cruz and H. Xiao, "The role of ontologies in data integration," *Journal of Engineering Intelligent Systems*, vol. 13, pp. 245–252, 2005.

[8] Esri, "ESRI Shapefile Technical Description," *Computational Statistics*, vol. 16, no. July, pp. 370–371, 1998.

[9] S. Software, "Feature Manipulation Engine," 2016, URL: http://www.safe.org/ [accessed: 2016-04-06].

[10] A. Buccella, A. Cechich, and N. R. Brisaboa, "Ontology-Based Data Integration Methods : A Framework for Comparison," *Revista Colombiana de Computación*, vol. 6, no. 1, 2005.

[11] INSPIRE Thematic Working Group Utility and governmental services, "D2.8.III.6 INSPIRE Data Specification on Utility and governmental services - Draft Technical Guidelines," INSPIRE Thematic Working Group, Tech. Rep. March, 2004.

[12] Anzlic, *One ANZ Foundation Spatial Data Framework*. ANZLIC, 2012, no. November.

[13] H. Wache, T. Scholz, H. Stieghahn, and B. Konig-Ries, "An integration method for the specification of rule-oriented mediators," in *Proceedings 1999 International Symposium on Database Applications in Non-Traditional Environments (DANTE'99) (Cat. No.PR00496)*, no. 01. IEEE Comput. Soc, 1999, pp. 109–112.

[14] EuroGEOSS, "EuroGEOSS," 2016, URL: http://www.eurogeoss.eu/default.aspx [accessed: 2016-04-06].

[15] P. A. Vretanos, "Web Feature Service Implementation Specification," Open Geospatial Consorium Inc., Tech. Rep., 2005.

[16] P. Staub, "A Model-Driven Web Feature Service for Enhanced Semantic Interoperability," *OSGeo Journal*, vol. 3, no. December, pp. 38–43, 2007.

[17] J. Partyka, N. Alipanah, L. Khan, B. Thuraisingham, and S. Shekhar, "Content-based ontology matching for GIS datasets," in *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems - GIS '08*, no. c. New York, New York, USA: ACM Press, 2008, pp. 1–4.

[18] M. Uschold and M. Gruninger, "Ontologies: principles, methods and applications," *The Knowledge Engineering Review*, vol. 11, no. 02, pp. 93–162, jul 1996.

[19] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, jun 1993.

[20] R. Megala and K. Nirmala, "Semantic Queries in Distributed Relational Database Using Global Ontology Construction." *ICTACT Journal on Soft Computing*, pp. 942–945, 2015.

[21] D. L. McGuinness and F. Van Harmelen, "OWL Web Ontology Language Overview," 2009, URL: https://www.w3.org/TR/owl-features/ [accessed: 2016-04-06].

[22] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. Mcllraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "OWL-S: Semantic Markup for Web Services," URL: http://www.w3.org/Submission/OWL-S/ [accessed: 2016-04-06].

[23] K. Stock, A. Robertson, and M. Small, "Representing OGC Geospatial Web Services in OWL-S Web Service Ontologies," *International Journal of Spatial data Infrastructures Research*, vol. 6, 2011.

[24] L. D. Ngan and R. Kanagasabai, "OWL-S Based Semantic Cloud Service Broker," *2012 IEEE 19th International Conference on Web Services*, pp. 560–567, 2012.

[25] P. Yue, L. Di, W. Yang, G. Yu, and P. Zhao, "Semantics-based automatic composition of geospatial Web service chains," *Computers and Geosciences*, vol. 33, no. 5, pp. 649–665, 2007.

[26] T. A. Farrag, A. I. Saleh, and H. A. Ali, "Toward SWSs discovery: Mapping from WSDL to OWL-S based on ontology search and standardization engine," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 5, pp. 1135–1147, 2013.

[27] P. Haase, I. Horrocks, and D. Hovland, "Optique System: Towards Ontology and Mapping Management in OBDA Solutions," *Second International Workshop on Debugging Ontologies and Ontology Mappings - WoDOOM13*, pp. 21–32, 2013.

[28] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo, "The MASTRO system for ontology-based data access," *Semantic Web*, vol. 2, no. 1, pp. 43–53, 2011.

[29] D. Lanti, M. Rezk, G. Xiao, and D. Calvanese, "The NPD benchmark: Reality check for OBDA systems," *Proc. 18th International Conference on Extending Database Technology (EDBT)*, pp. 617–628, 2015.

[30] N. Antonioli, F. Castanò, C. Civili, S. Coletta, S. Grossi, D. Lembo, M. Lenzerini, A. Poggi, D. F. Savo, and E. Virardi, "Ontology-Based Data Access: The Experience at the Italian Department of Treasury," *CAiSE Industrial Track*, vol. 1017, pp. 9–16, 2013.

[31] X. Yang, W. Cui, Z. Liu, and F. Ouyang, "Study on uncertainty of geospatial semantic Web services composition based on broker approach and Bayesian networks," in *Geoinformatics 2008 and Joint Conference on GIS and Built Environment: Geo-Simulation and Virtual GIS*

 ISBN: 978-1-61208-469-5

Environments, L. Liu, X. Li, K. Liu, X. Zhang, and A. Chen, Eds., vol. 7143, oct 2008, pp. 714 305–714 305–8.

[32] Anzlic, "ANZLIC - the Spatial Information Council is the peak intergovernmental organisation providing leadership in the collection, management and use of spatial information in Australia and New Zealand." 2016, URL: http://www.anzlic.gov.au/ [accessed: 2016-04-06].

[33] "Protégé," 2016, URL: http://protege.stanford.edu/ [accessed: 2016-04-06].

[34] S. J. D. Cox, "OWL representation of ISO 19107 (Geographic Information - Spatial Schema)," 2015, URL: http://def.seegrid.csiro.au/isotc211/iso19107/2003/geometry [accessed: 2016-04-06].

[35] Rhizomik, "ReDeFer," 2016, URL: http://rhizomik.net/html/redefer/#XSD2OWL [accessed: 2016-04-06].

[36] Brishniz, "XML2OWL Demonstration Platform," 2016, URL: http://xml2owl.sourceforge.net/index.php [accessed: 2016-04-06].

[37] Incunabulum, "XsdImport - Convert XSD schemas to OWL," 2016, URL: http://www.incunabulum.de/projects/it/xsdimport/ [accessed: 2016-04-06].

[38] "Python," 2016, URL: https://www.python.org/ [accessed: 2016-04-06].

[39] "Django: The web framework for perfectionists with deadlines." 2016, URL: https://www.djangoproject.com/ [accessed: 2016-04-06].

 ISBN: 978-1-61208-469-5