A Service Oriented Architecture for Extracting and Extending Sub-Ontologies in the Semantic Grid

Andrew Flahive[1], Wenny Rahayu[1], David Taniar[2],
Bernady O. Apduhan[3], Carlo Wouters[1] and Tharam Dillon[4]

1. La Trobe University, Australia
Email: {A.Flahive, W.Rahayu, C.Wouters}@latrobe.edu.au
2. Monash University, Australia
Email: David.Taniar@infotech.monash.edu.au
3. Kyushu Sangyo University, Japan
Email: bob@is.kyusan-u.ac.jp
4. University of Technology Sydney, Australia
Email: Tharam@it.uts.edu.au

Abstract

This paper presents a Service Oriented Architecture (SOA) approach to a distributed framework for reusing, extracting and extending (tailoring) large domain ontologies in the Semantic Grid Environment. The conceptual level of the framework describes how sub-ontologies are tailored while the architectural level of the framework describes the components of the framework that allows the tailoring process happen in the Semantic Grid Environment. A prototype of the framework and a complexity evaluation measure are also provided.

1 Introduction

Ontologies were designed to maintain the expressiveness[1] of the domain they capture rather than for ease of use. It is for this reason that many are looking for more efficient methods for ontology reuse, extraction and extension, what we call *tailoring*[2]. There are three main reasons why ontologies are required to be tailored; 1. to enable faster access to the most relevant information, 2. to assist in finding new knowledge from within large domain ontologies and, 3. to assist in ontology creation.

Areas such as Life Sciences [3] and Particle Physics [4] are some of the main motivators in large scale knowledge acquisition. They often require access to large data repositories so that new knowledge can be discovered and extracted. E-science is a collaborative effort that covers many areas of science that deal with very large data collections and very large scale computing resources[5].

Ontology creation is a large task and any assistance that can be given will drastically reduce their creation time. Creating an ontology from scratch has many benefits, such as designing the ontology for a particular use or application rather than create it for general use. It allows the creator to cut corners to make the ontology more efficient and more directed toward the target application. Creating ontology from scratch often takes a long time and is very tedious.

This paper introduces a Service Oriented Architecture approach for ontology extraction and extension. This will allow the previous work in the area to reach many more people by using simple, easy to use Web Services.

2 Background and Related work

This section provides a brief background into the Semantic Grid and how it differs from the Computational Grid. This section also introduces the concept of ontology tailoring before describing some related work.

2.1 The Grid vs The Semantic Grid

The Grid is "...flexible, secure and coordinated resource sharing among dynamic collections of individuals, institutions and resources - what we refer to as Virtual Organizations"[6]. The Grid supports on demand Virtual Organizations (VOs) for coordinated problem solving on a global scale[7]. It does this by combining many distributed, autonomous computing resources and allowing consumers access to these resources and harness their ability.

The Semantic Grid is about facilitating the interchange and integration of heterogeneous information[8]. It makes an improvement on the Grid by allowing for efficient knowledge management among its services, better enabling computers and people to work in together.

2.2 Ontology Tailoring

An ontology is necessary to capture the expressive power that is needed for modeling and reasoning with knowledge. Ontologies are ideal for associating various pieces of information through its many complex entities and relationships.

Large domain ontologies can contain many millions of concepts and relationships and may be ever expanding as new information is gathered. Some of the larger ontologies include the Unified Medical Language System (UMLS) [9] ontology with over 1 million biomedical concepts and 5 million concept names.

Deriving tailored ontologies from large base ontologies involves reusing, extracting and information and enables individuals to obtain only specific parts of the ontology for their intended use. Smaller ontologies, rather than whole base ontologies, are much better suited to the user [10, 11].

Ontology tailoring, is computationally expensive, in part because of the size of the ontologies and also partly because of the complexity of the requirements of the user. Our earlier work in the area [12, 13, 14, 15, 16], *Materialized Ontology View Extractor (MOVE)*, has addressed this problem, to a degree, by proposing a distributed architecture for the extraction/optimization of a sub-ontology from a large base ontology.

2.3 Related Work

The myGrid project[17] has developed a middleware framework that aims to assist researchers in the drug discovery process. It gathers together rich, semantically enabled collections of services so that applications can be built for the non-human part of the scientific drug discovery process.

Ontology Segmentation is another approach that the authors of the article [18] have taken. The authors understand the difficulties that large ontologies face and attempt to solve these through segmentation. By creating smaller ontologies they are easier to classify, traverse and use day to day activities.

3 The Proposed Framework

The proposed framework is described in this section on two different levels; the *conceptual level* and the *architectural level*. The conceptual level of the framework defines the underlying process that takes place to extract a complete and valid sub-ontology while the architectural level of the framework defines the components that allow the process to take place efficiently within the Semantic Grid Environment.

3.1 The Conceptual Level

The conceptual level of the framework outlines process behind the sub-ontology extraction process. The means of extracting a complete and valid sub-ontology ready to be reused are discussed and defined before the extension to the reusable sub-ontology.

3.1.1 Reuse and Extract

This section provides a formal definition of the reuse and extract methodology illustrated in Figure 1. The large Domain Ontology is reused by extracting parts of it into smaller sub-ontologies which are complete and valid ontologies on their own. Figure 1 shows that the large Domain Ontology has been reused three times. Subsumption with restriction has been applied to the large domain ontology resulting in derived and restricted sub-ontologies.

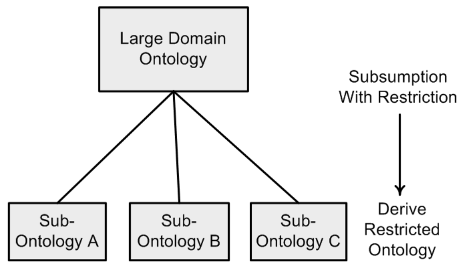

Figure 1. Reuse and Extract

This methodology has been adapted from the initial formalization found in [19] and also in our earlier work [13, 12]. For this paper we extend the formalization to fit into our proposed framework in the Semantic Grid.

Definition 1 *An ontology O is made up of a set of Concepts (C), Concept Properties (P), Property Mapping (T) and Relationships between the Concepts (R).*

$$O = \{C, P, T, R\}$$

This ontology definition describes an ontology in terms of the entities that are found within it.

Definition 2 *A valid ontology O is defined as an ontology that can be viewed as an interconnected graph.*

A complete ontology is one that conforms to a set of common laws, like aggregation or subsumption.

Definition 3 *A complete ontology O is defined as an ontology that is semantically perfect and is accepted based on the meaning of its contents rather than its interconnectivity.*

A $complete$ and $valid$ ontology therefore, conforms to all of the rules regarding the meaning of its contents as well as the interconnectivity of its elements.

A subset of an ontology is made up of a number of Concepts, Properties, Property Mappings and Relationships that exist within a given ontology O.

Definition 4 *A Subset S of an Ontology O is a part or section of that ontology, containing a sample of its Concepts, Properties, Property Mappings and Relationships. 1. Let $C^{\#} \subseteq C$, 2. Let $P^{\#} \subseteq P$, 3. Let $T^{\#} \subseteq T$, 4. Let $R^{\#} \subseteq R$,*

$$S = \{C^{\#}, P^{\#}, T^{\#}, R^{\#}\}$$

A Sub-set S of an Ontology O is here on defined as:

$$S \subseteq O$$

It should be noted that, as there are no rules governing the validity of a sub-set of an ontology, a subset does not necessarily conform to the constraints of a $complete\ and\ valid$ ontology. When attempting to reuse a domain ontology it is important that the extracted subontology is $complete\ and\ valid$ and an ontology in its own right.

Definition 5 *Let O' be complete and valid sub-ontology of ontology O 1. $C' \subseteq C$, 2. $P' \subseteq P$, 3. $T' \subseteq T$, 4. $R' \subseteq R$,*

$$O' = \{C', P', T', R'\}$$

A Sub-ontology O' of Ontology O is therefore defined as

$$O' \Subset O$$

This definition of a sub-ontology means that it is itself a $complete\ and\ valid$ ontology on its own. It also means that its entities are subsets of the original ontology O.

Proposition 1 *A subset S of ontology O can only be equivalent to sub-ontology O' of ontology O if, given:*

$$O' = \{C', P', T', R'\}\ and$$

$$S = \{C^{\#}, P^{\#}, T^{\#}, R^{\#}\}$$

with $C' = \{C^{\#} \wedge P' = P^{\#} \wedge T' = T^{\#} \wedge R' = R^{\#}\}$

To reuse various entities within an ontology an indication of which entities are required would have to be provided. The process of labeling the ontology is selecting what should and should not be reused. Reusing is specific to labeling. The process of reusing is defined as follows:

Definition 6 *A labeling of an ontology defines the set of concepts, properties, property mappings and relationships that should or should not be reused in the resulting subontology.*

1. 'Selected' indicates that the entity is required to be reused and it will be extracted as part of the sub-ontology.

2. 'Deselected' indicates that the entity is not allowed to be apart of the extracted sub-ontology.

3. 'Undecided' indicates that the entity is not required at this time and can be changed to 'selected' if it is required to form a complete and valid sub-ontology.

$$Let\ L\ be\ the\ set\ of\ Labels$$

$$L = \{`deselected', `deselected', `undecided'\}$$

Definition 7 *Let σ define when a set of entities or an entire ontology has had labeling applied to each of their elements. A set of entities of an Ontology O are said to be labeled when:*

$$\sigma(O) = \{\sigma(C), \sigma(P), \sigma(T), \sigma(R)\}$$

Where certain labeled entities need to be described this notation is used:

Notation 1 *Let* $\sigma_S \equiv all\ `selected'\ entities$ *Let* $\sigma_D \equiv all\ `deselected'\ entities$ *and Let* $\sigma_U \equiv all\ `undecided'\ entities$

The purpose of an Optimization Scheme (OS)[12, 13] is to check whether the current labeled set σ requires any further entities to become a complete and valid ontology $\rightarrow$ a sub-ontology. Where certain other entities are required the OS turns the label of the entity from *'undecided'* to *'selected'*. If the particular label of the entity is *'deselected'* then the OS must try to find another solution. If no other solution is possible σ cannot become a sub-ontology.

3.1.2 Reuse, Extract and Extend

Continuing from the expression for the reuse and extraction process defined in the previous section, we propose a methodology to extend the sub-ontology whilst maintaining a connection to the original domain ontology. As illustrated in Figure 2 Sub-Ontology A and Sub-Ontology B have been extracted from the Domain Ontology and extended with new concepts. This extension is important for the new intended use of the Sub-Ontology.

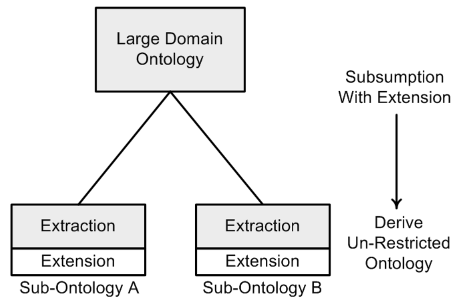

Figure 2. Reuse, Extract and Extend

This section extends this notion of sub-ontology to include new features relevant to this sub-domain. The new features are in terms of Concepts, Properties, Property Mappings and Relationships known as entities. The new features contain a set of entities that are not necessarily related to each other.

Definition 8 *Let O^z be defined as a new set of entities. If adding a new set of entities $O^z \neq Null$. However, C^z, P^z, T^z and R^z may be equal to Null as long as one of them does not.*

$$O^z = \{C^z, P^z, T^z, R^z\}$$

O^z is a list of entities that are used to extend O'. These entities represent the new information that the organization is adding, perhaps to describe new products or new services that are now available.

Definition 9 *Let Q be defined as a new set of entities that combine the extracted sub-ontology O' and the new set of entities O^z.*

$$Q = O^z \cup O'$$

Q is not a valid sub-ontology and hence not a $complete\ and\ valid$ ontology. Note also that Q is not a sub-ontology of O as there exists entities in Q that do not exist in the original ontology O.

As O' has now changed from a complete and valid sub-ontology to Q, which may or may not be a valid ontology. Q must be validated using an Optimization Scheme before it can be used as a $complete\ and\ valid$ ontology.

3.2 The Architectural Framework

The *Distributed Ontology Framework (DOF)* is an attempt to define the requirements of knowledge engineering system that tailors large ontologies by efficiently making use of Semantic Grid resources.

Figure 3 shows an example of some of the components/entities that may make up the distributed framework in a *Service Oriented Architecture*.

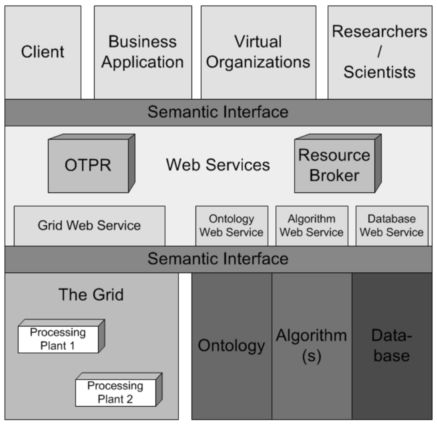

Figure 3. The Service Oriented Architecture of the Framework

The main components are: User, Ontology Tailoring Program Resource (OTPR), Processing Plant, Ontology, Tailoring Algorithms, and the Resource Broker. All of these components interact with each other to perform the task of tailoring large ontologies based on a set of initial user requirements.

4 Implementation

A Service-Oriented Architecture was chosen due to the distributed nature of the components within the framework. The prototype presented in this section demonstrates the validity of such a framework within the Semantic Grid Environment. This section also discusses the limitation of this Web services prototype and why the framework requires the addition of some Semantic Grid services in order to have full functionality and impact.

4.1 Implementation Environment

The implementation environment was Java (J2EE 1.4) and the platform was Windows XP, including SP 2 on each of the four machines used. These four machines hosted some of the Web services and demonstrate the distributed nature of the components.

Apache Web Server (Apache 2.0.54) and Tomcat (Version 5.5.9) was installed on each of the four machines to deploy and test the framework Web Service components. Axis (Version 1.2.1) was also used in the prototype to speed up the Web Service deployment process.

A MySQL database (MySQL 4.1.12) was used to house the ontology for testing purposes. To enable a connection to be made from the Java Web Services top the database the Java connector (Version 3.1.10) was installed. For the communication between the Web Services SOAP (Version 2.3.1) was used. XML was used to transport the messages between the Web Services in SOAP.

4.2 Components

The components/entities of the distributed framework are: User(Client), Algorithm Server, Ontology Server, Processing Plant and the Ontology Tailoring Program Resource (OTPR).

The components of the distributed ontology framework that are discussed in this section are: The Algorithm Service, the Ontology Service, the Client(User), the Resource Broker and the OTPR.

4.3 Algorithm Server Web Service

The duty of the Algorithm Server is to provide the Algorithms when requested by the OTPR. In this Service, the algorithms are stored in XML as strings. Shown below, is an example of the Algorithm Server as deployed as a Web Service.

```
public class AlgorithmServer3
{
   public String getAlgorithm3(String name)
   {
      if( name.equals("all") )
            return "<equation name=\"RCOS\" al_type=\"Simple\">
</equation>
<equation name=\"SCOS\" al_type=\"Simple\">
</equation>
<equation name=\"TSOS\" al_type=\"Complex\">
</equation>";

       if( name.equals("RCOS") || name.equals("rcos"))
return "<equation name=\"RCOS\" al_type=\"Simple\">
</equation>";

       if( name.equals("SCOS") || name.equals("scos"))
return "<equation name=\"SCOS\" al_type=\"Simple\">
</equation>";

       if( name.equals("TSOS") || name.equals("tsos"))
return "<equation name=\"TSOS\" al_type=\"Complex\">
</equation>";
       else
          return "Cannot find the " + name + " Algorithms
          try RCOS or SCOS or TSOS or all";
   }
}
```

In the above code there are three algorithms stored, RCOS, SOCS and TSOS (A more detailed description of these algorithms can be found in our previous work [19, 12]. When this Web Service is invoked by the OTPR, it expects an indication of the algorithm(s) required. Based on the contents of the request the correct algorithms are returned.

The algorithms have been stored as XML data. This XML data is passed straight back to the OTPR (the requesting entity) as it is in the required format. XML is used as the data transfer language throughout the Web Services Implementation of the Distributed Ontology Framework as it was found that this is the most efficient and scalable method.

4.4 Ontology Server Web Service

The role of the Ontology Server Web Service is to provide certain parts of an ontology to the OTPR. If the ontology is segmented and distributed over the Semantic Grid Environment remote commands make it easier to obtain the parts of the ontology required.

A selection of the upper cyc ontology[20] has been selected as data for the ontology database. This portion of the ontology contains about 3000 concepts and over 7000 relationships. This ontology has been modified slightly so that it can fit into the current implementation of our ontology database.

The Ontology Server Web Service is given below:

```
...
try {
// Set up the Database Connection
Class.forName("com.mysql.jdbc.Driver").newInstance();
  con = DriverManager.getConnection("jdbc:
  mysql:///ontologydb", "MyUsername", "MyPassword");

// Set up the part of the Ontology to Retrieve
  if( instruction.equals("basicGraph") )
  {
st = con.createStatement();
    rs = st.executeQuery("select ToConceptID,
         FromConceptID from relationshiptable;");

   // Get the parts of the ontology, put them
   // in XML and return them to the OTPR
    while(rs.next())
    {
     firstcolumn = rs.getString(1);
secondcolumn = rs.getString(2);
allfirstcolumn = allfirstcolumn + "||" + firstcolumn;
    allsecondcolumn = allsecondcolumn + "||" + secondcolumn;
    }
    return allfirstcolumn + allsecondcolumn;
  }
  else
    return "Cannot find the " + instruction + " Instruction";
...
}
```

In this example, the Ontology Service expects the command ‘basicGraph’. Once it receives this request, it queries the ontology database using an appropriate query, collects the information (in this case two columns of data) and returns them to the requesting entity (OTPR).

4.5 The OTPR Web Service

The prototype of the OTPR is presented below. Its main role is to manage the user request and handle the communication between the various services. The OTPR manages all of the other details such as, establishing a connection to the various services and tailoring the results for the User. Below is a section of the OTPR Web Service.

```
...
         // Build the call to the Algorithm Service.
         Call call = new Call();
         call.setTargetObjectURI("urn:Algorithm");
         call.setMethodName("getAlgorithm");
         call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);

         // Invoke the call to Algorithms Service.
            resp = call.invoke(algurl, "");

         //Built the call to the Ontology Service
         Call call2 = new Call();
         call2.setTargetObjectURI("urn:Ontology");
         call2.setMethodName("getOntology");
         call2.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
         params2.add(new Parameter("basicGraph"));
```

```
        // Invoke the call to Ontology Service.
           resp2 = call2.invoke(onturl, "");

        // Check the response and decide processing requirements.
           Parameter ret2 = resp2.getReturnValue();

  if((NumComplex == 0) || ((isSmall > 0) && (NumComplex < 1)))
   tailored_ontology = TailorOntology(Locally);
   else ProcessRemotely = 1;

if((ProcessRemotely == 1) && ((TimeLimit > 0) || (Budget > 0)))
tailored_ontology = TailorOntology(RB);
else
if (ProcessRemotely == 1)
tailored_ontology = TailorOntology(PP,RB); //PP or RB

  // Return the Tailored Ontology back to the user.
        return "The OTPR has successfully tailored
         your ontology: " + tailored_ontology;
...
```

In this example the OTPR is invoked by the User and expects two things; the location of the Algorithm Service with an indication of the algorithm to select and the location of the Ontology Service.

The OTPR Web Service then invokes two Web Services; the Algorithm Service and the Ontology Service. Based on the user requirements the correct algorithm is obtained. Depending on the algorithms the OTPR collects, various pieces of the ontology are requested and tailored. Once finished the OTPR sends the tailored ontology back to the User.

4.6 Client or User

The role of the User is to invoke a call to the OTPR (or Resource Broker). The user passes the location of the Algorithm and Ontology Services to the OTPR as well as the algorithms to select. At the end of the process the User accepts a tailored ontology from the OTPR.

The code below shows the User invoking the OTPR or Resource Broker based on a number of Runtime decisions.

```
// Set up the call
  algurl = new URL("http://localhost:8080/axis/AlgorithmServer3.jws");

OTPR = CheckKnownOTPRs();

  If ( OTPR == TRUE );
  {
    // Build the call to the OTPR.
   Call call = new Call();
   call.setTargetObjectURI("urn:OTPR");
   call.setMethodName("processOntology");
   call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
   resp = call.invoke(otprURL, "");
  }
  else
  {
    // Build the call to a Resource Broker.
   Call call = new Call();
   call.setTargetObjectURI("urn:Resource Broker");
   call.setMethodName("findOTPR");
   call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);

   resp = call.invoke(RB_URL, "");
  }
  Tailored_Ontology = resp.getReturnedOntology();
  Output(Tailored_Ontology);
```

5 Evaluation and Discussion

The proposed framework will not only assist with extracting sub-ontologies faster and cheaper but will also allow them to be extended and reused again if required. As the tailoring process has been accelerated, re-tailoring an ontology is not a problem as the labeling can be altered and the process restarted. In previous efforts much more emphasis was put on caching extracted sub-ontologies, for use elsewhere, as the extraction process took a long time. Now, due to speed and efficient methods, more effort can be put into knowledge discovery and ontology development rather than developing methods for caching extracted sub-ontologies.

5.1 Evaluation

A qualitative evaluation is provided in this section, with details as to how the performance of the framework is expected to behave once the system is fully functional. It is important to discuss these issues now so as avoid possible performance degradation further down the road.

5.1.1 Resource Utilization

In this evaluation of ontology reuse, extraction and extension, the amount of time taken by each of the resources, in this process, is investigated. This evaluation aims to demonstrate the abilities and weaknesses of the proposed framework. Then the evaluation will then drill down into one of the steps in the process to analyze the internal communication.

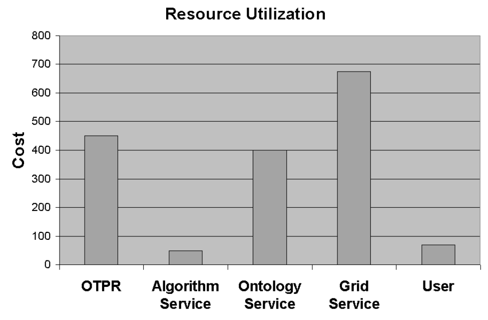

Figure 4. Resource Utilization Cost

From these parameters a complexity measure equation has been developed for each of the resources shown in Figure 4. This measurement takes into consideration the complexity of the algorithms used in the Optimization Schemes as well as the size of the ontology. This graph shows that for even medium sized ontologies and average complex algorithms the tailoring process is dominated by the OTPR, the Ontology server and the Processing Plants (Grid Service).

5.1.2 Communication Costs

Due to the large costs associated with the OTPR, Ontology Service and the Grid Service, a more detailed evaluation took place of these three resources. In particular the communication cost of the messages that are sent between these resources was investigated.

In the first case A an initial 50 entities were provided to the processing plant for tailoring. Due to the low complexity of the algorithms and the quality of the initial set of $Labeled$ entities, no further entities were required to be retrieved from the ontology server. Hence, a minimal amount of communication overhead is expected.

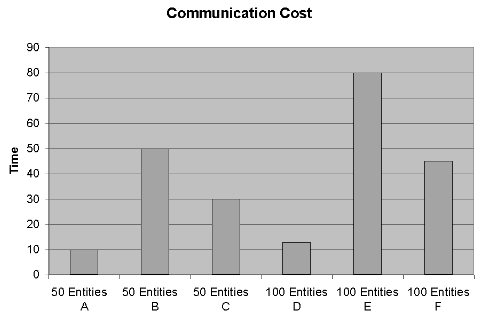

Figure 5. Communication Costs

In the second case (Case B) the same 50 entities were provided to the processing plant but this time several complex algorithms were applied. The Grid Service required further information about the $Labeled$ entities and had to consult the Ontology Server through the OTPR many times. This added a lot of overhead.

Similar observations are expected when a larger $Labeled$ set of entities is used. Case D E and F show corresponding communication costs for 100 entities in the initial $Labeled$ set.

The communication cost is made up of several parts, initial opening of connection to the Ontology server, the requesting of the initial number of entities from the Ontology Server, the sending back of this information to the OTPR, the initialization of the Processing Plant(s) within the Grid Service, the sending of the initial $Labeled$ set of entities from the OTPR to the processing plant and the sending of the tailored ontology back to the User. This is the minimum communication cost associated with the process and is largely due to the size of the ontology and the number of $Labeled$ entities.

5.2 Limitations

The main limitation of this current implementation is that no processing can currently be completed by the OTPR. The OTPR needs to have ontology processing capabilities included as well as the option to employ a Processing Plant. These components need to be created as Semantic Grid Services and connected to using standard Grid Service methods.

The main task of the resource broker is to match available resources with the user's demand or specifications, and to manage the tailoring work to be completed on-time and on-budget. It is outside of the scope of this prototype to manage the processing to this degree. There are many such implementations of a resource broker available[21, 22, 23]. Only a simple version of a Resource Broker is created here for testing purposes.

The current implementation can go as far as to collect all of the relevant information from the various servers ready for the OTPR to process them. Executing the code from the algorithms on the OTPR is a challenging task and will require further planning.

6 Conclusion and Future Work

To enable researchers, scientists and engineers to better gain new knowledge from large ontologies, a framework has been presented and developed in this paper. The framework exploits the computational resources of the Semantic Grid and the ease of Web Services to allow users to tailor large ontologies. A Service Oriented approach was taken for the implementation of several components in the framework prototype. Web Services were found to be ideal for the algorithm service and the ontology service but several semantic grid services will have to be developed to take advantage of the hardware based resources.

The Service Oriented components of the framework were implemented and show that the Semantic Grid is an ideal environment for large scale knowledge acquisition. Several components were implemented as Web Services that will interact with Grid Services to perform the main tailoring tasks.

The Grid Services implementation is the next phase of this work. This will efficiently make use of available computational resources to tailor the large ontologies. The Grid Services can then interact easily with the Web Services, presented in this paper to provide users with an easily accessible resource.

7 Acknowledgment

The authors of this paper would like to thank the Victorian Partnership of Advanced Computing (VPAC) for their financial support in the early part of this research. Expertise Grant Round 5, reference number EPPNLA090.2003.

References

[1] J. Heflin, "Owl web ontology language, use cases and requirements, w3c recommendation 10 february 2004," http://www.w3.org/TR/webont-req/.

[2] A. Flahive, W. Rahayu, D. Taniar, and B. Apduhan, "A distributed ontology framework for the grid," in *Parallel and Distributed Computing: Applications and Technologies (PDCAT)*, vol. LNCS 3320. Springer-Verlag, 2004, pp. 68–71.

[3] K.-H. Cheung, K. Y. Yip, A. Smith, R. Deknikker, A. Masiar, and M. Gerstein, "Yeasthub: a semantic web use case for integrating data in the life sciences domain," *Bioinformatics*, vol. 21 Suppl 1, pp. i85–i96, June 2005.

[4] J. L. Bell, "A specialized data management system for parallel execution of particle physics codes," in *Proceedings of the 1988 ACM SIGMOD*. USA: ACM Press, 1988, pp. 277–285.

[5] Research Councils UK, "e-science," http://www.rcuk.ac.uk/escience/, 2006.

[6] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, no. 3, pp. 200–222, 2001.

[7] C. Goble and D. D. Roure, "The grid: An application of the semantic web," *CACM*, vol. 31, no. 4, pp. 65–70, December 2002.

[8] D. D. Roure, N. Jennings, and N. Shadbolt, "The semantic grid: A future e-science infrastructure," in *Grid Computing*, F. Berman, A. Hey, and G. Fox, Eds. England: John Wiley, 2003, pp. 437–470.

[9] United States National Library of Medicine, "Unified medical language system," 2006, http://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html.

[10] C. Wouters, T. Dillon, W. Rahayu, and E. Chang, "A practical walkthrough of the ontology derivation rules," *DEXA2002*, pp. 259–268, 2002.

[11] P. Spyns, R. Meersman, and J. Mustafa, "Data modelling versus ontology engineering," *SIGMOD*, 2002.

[12] C. Wouters, T. Dillon, W. Rahayu, E. Chang, and R. Meersman, "A practical approach to the derivation of materialized ontology views," in *Web Information Systems*, D. Taniar and W. Rahayu, Eds. Idea Group Publishing, 2004, pp. 191–226.

[13] C. Wouters, T. Dillon, W. Rahayu, E. Chang, , and R. Meersman, "Ontologies on the move," in *Database Systems for Advanced Application (DASFAA 2004)*, ser. LNCS. Springer Verlag, 2004, pp. 812 – 823.

[14] M. Bhatt, A. Flahive, C. Wouters, W. Rahayu, and D. Taniar, "A distributed approach to sub-ontology extraction," in *Advanced Information Networking and Applications (AINA'04)*, Japan, 2004, pp. 636–641.

[15] M. Bhatt, C. Wouters, A. Flahive, W. Rahayu, and D. Taniar, "Semantic completeness in sub-ontology extraction using distributed methods," in *Computational Science and its Applications (ICCSA04), LNCS*. Italy: Springer-Verlag, 2004, pp. 508–517.

[16] M. Bhatt, A. Flahive, C. Wouters, W. Rahayu, and D. Taniar, "Move: A distributed framework for materialized ontology view extraction," *Algorithmica*, vol. Volume 45 Issue 3, pp. 457 – 481, July 2006.

[17] R. Stevens, R. Mcentire, C. Goble, M. Greenwood, J. Zhao, A. Wipat, and P. Li, "mygrid and the drug discovery process," *Drug Discovery Today: BIOSILICO*, vol. 2, no. 4, pp. 140–148, July 2004.

[18] J. Seidenberg and A. Rector, "Web ontology segmentation: analysis, classification and use," in *WWW '06: Proceedings of the 15th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2006, pp. 13–22.

[19] C. Wouters, "A formalization and application of ontology extraction," Ph.D. dissertation, La Trobe University, Melbourne, Australia, 2005.

[20] Cycorp Inc., "Creators of the cyc knowledge base," *CYCORP Website*, 2003, http://www.cyc.com/.

[21] B. Lesyng, P. Baia, and D. Erwin, "Eurogrid: European computational grid testbed," *J. Parallel Distrib. Comput.*, vol. 63, no. 5, pp. 590–596, 2003.

[22] J. Brooke, D. Fellows, K. L. Garwood, and C. A. Goble, "Semantic matching of grid resource descriptions," in *European Across-Grids Conference*, Cyprus 2004, pp. 240–249.

[23] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid," in *High Performance Computing in Asia-Pacific Region*. China: IEEE CS Press, 2000, pp. 283–289.