ABSTRACT
The main objective of this research work is to extend the capability of Globus Toolkit (GT) to support semantic description and discovery of Grid Services. We proposed Protégé Enabled Globus toolkit(PEG) in which we have integrated GT with Protégé editor to support globus user for semantic descriptions of Grid services and the Algernon inference engine is used to interact with the OWLS descriptions. We have also proposed a new algorithm called parameter matchmaking algorithm that computes various degrees of matching of advertised service descriptions with that of the requested ones based on the Input, Output and Functionality (IOF) parameters. On the contrary to algorithms that return only success or fail, ranked degrees of match obtained from our proposed algorithm provide better precision against the selection of a service among a large set of services. The proposed algorithm is tested successfully in PEG toolkit for the semantic discovery of grid services.

KEY WORDS

1. Introduction
Grid Technologies support sharing and coordinated use of diverse resources, from geographically and organizationally distributed components, of virtual computing systems that are sufficiently integrated to deliver desired Quality of Services [1, 2]. The grid computing infrastructure defined in [1, 2] is only a part of much larger picture that also includes information handling and support for knowledge processing within the distributed scientific process. This broader view is adopted for semantic grid which can be described as an extension of the current grid where information and services are given well defined meaning, better enabling computers and people to work in cooperation [3, 4]. The semantic grid is a service oriented architecture in which entities provide services to one another under various forms of contract. In such an environment, it is essential to facilitate the user for easier discovery of the available services.

The rest of the paper is organized as follows:- Section 2 discusses the related work in Grid service discovery and distinguish our work with those related work. The section 3 deals with basic concepts of semantic grid services using PEG. The proposed architecture for semantic grid service using PEG is explained in section 4. The section 5 describes the Parameter Matchmaking Algorithm implemented in this paper. The implementation details and experimental results followed by brief discussion on the results are described in the section 6 and section 7 concludes the paper by aggregating the work done in this paper.

2. Related Work
Searching by traditional search engines based on keywords [5] has its own problems. As a result they do not check semantic of search objects and simply treat them as character strings. A lot of irrelevant information will be returned to the user as long the keywords appear somewhere in their files.

The literature [6] proposes a meta search engine called “guided google”, that is built using the Google web services. This search engine guides and allows the user to view the search results with different perspectives, which achieved through simple manipulation and automation of Google functions. However, the functionalities provided in this engine are based on “combinational keyword
search” and it neither supports semantic description nor performs semantic search.

The literature [7] addresses the problem of resource description in the context of resource broker to broker for resources described by several Grid middlewares including Unicore.

In this research work, we propose a semantic grid architecture using PEG that addresses the issue of semantic description through OWLS plugin and discovery of services. The PEG allows the service provider for providing semantic description of grid services using OWLS editor that comes as a plugin to protégé ontology editor. The Parameter Matchmaking Algorithm proposed in this paper compares the IOF of advertised and requested service and determines the degrees of match. This degrees of match reveals how similar they are. Algernon inference engine [8] is used to retrieve IOF parameters from the advertised service ontology in PEG. We also propose an semantic grid architecture in which protégé editor is integrated with Globus middleware making possible for semantic descriptions of Grid services.

3. Semantic Grid Services using PEG

The PEG addresses the demands of a single toolkit to build Grid infrastructure as well as for semantic description and representation of services. Currently, the concept of ontology is widely used for conceptual representation of a particular domain and Web Ontology Language (OWL) is used to develop ontology as a concept. Ontologies are used to capture knowledge of a domain of interest. Ontology describes the concepts in the domain and also the relationships that hold between those concepts [9]. The most recent development in standard ontology language is OWL from the World Wide Web Consortium. It makes it possible to describe concepts and it also provides a richer set of operators such as and, or and negation. It is based on different logical models which make it possible for concepts to be defined as well as described [10]. Further, the logical model allows the use of reasoner which can check whether all of the statements and definitions in the ontology are mutually consistent or not and it can also recognize which concept falls under which definition. Protégé editor is an integrated software tool used by system developers and domain experts to develop knowledge based systems [11]. This tool is widely used to create ontology in many applications. Protégé editor has Algernon inference engine has plugin which facilitates direct interaction with Protégé knowledge bases (K Bs) and it supports access to multiple concurrent K Bs. Algernon commands not only retrieve and store slot values, but can also modify the ontology by executing Algernon queries.

In PEG, the Protégé editor along with OWLS plugin is integrated with GT to address the demand of single toolkit for semantic description and representation of services by creating service ontology and its capability is extended to enable semantic description and representation of services by creating service ontology. In this paper, we propose a five layered architecture using PEG as middleware for semantic description and discovery of services.

4. Layered Architecture of Semantic Grid using PEG

The five layered architecture proposed for semantic grid services is shown in Figure 1. Each layer shares the behavior of the underlying component layers and the same is explained below.

Fabric layer
The Fabric layer deals with the resources available in grid environment and defines the interface to local resources, which may be shared. This includes computational resources, data storage, networks, catalogs, software modules, and other system resources.

Grid Middleware Services
This layer incorporates Grid Middleware and we use PEG as Grid Middleware in this research work. It also consists of required protocols for Authentication and Authorization which are implemented using Grid Security Infrastructure provided by PEG.

Knowledge services layer
Running on top of the high level grid service layers, the knowledge service layer can provide knowledge discovery from a huge amount of data. This layer is
domain oriented and usually consists of service ontology built using protégé editor. The parameter matchmaking algorithm proposed in this paper is implemented in this layer that performs matchmaking of services based on IOF parameters.

**Application layer**
The application layer enables the use of resources in a grid environment through various collaboration and resource access protocols. The semantic portlet present at this layer enables the service provider to register the service into the MDS registry and it prompts the provider to describe the service semantically using Protégé editor. The portlet also enables the service requester to submit the query and semantic retrieval of information from the service ontology using the proposed matchmaking algorithm. In addition to that, this layer may also consist of various application portlets to use grid resources.

4.1. Semantic Component

The semantic component proposed in the knowledge layer enables the service provider to describe the service semantically. Also, the component accomplishes semantic discovery of services. The OWLS editor that comes as a plugin to protégé editor allows the service provider to semantically describe the characteristics of the service including Inputs, Outputs, Functionalities (IOF). The discovery module gets the description of the requested service in the form of Inputs, Outputs, and Functionality. Further, it uses Algernon inference engine to interact with the OWLS files and retrieves IOF of the advertised services. It then semantically compares these parameters with that of the requested ones and obtains the suitable services using Parameter Matchmaking Algorithm. The semantic comparison in our context refers determining the similarity between requested parameter and the advertised. This is done using background ontology where possible input and output for a service is modeled. For Ex: if the requested parameter is “float” then the advertised is “float”, then semantic comparison infers that there exist a slight similarity between these parameters as “float” may also be served for “int” though they don’t exactly matches. We use Algernon inference engine to infer this information from the background ontology. With this information, the algorithm computes respective degrees of match between the advertised service ontology and the requested service on the basis of IOF parameters.

5. Parameter Matchmaking Algorithm

Matchmaking refers to capability matching which means to compare the requested service description with the advertised service descriptions [12]. In this paper, we use IOF to express the capability of a service. The goal of this capability comparison is to obtain information on how similar they are [12]. This degree of similarity is used to determine degrees of match between the advertised services and the requested capabilities. Comparing the requested service requirements with the advertised service descriptions takes all the inputs and the outputs into account [13]. In this research work, the proposed algorithm computes various matching degrees of service advertisement (A) and request (R) by successively applying different filters. The comparison is based on three parameters of the service namely the Inputs, Outputs and Functionalities (IOF). The service ontology that clearly describes IOF of the service is created using protégé editor of PEG to enable effective matchmaking of services. The algorithm semantically compares the IOF of the requested services with that of the advertised ones and computes various degrees of matches as listed below:-

**Exact Match**: Here the advertised IOF of the service are exactly matches with that of requested service. We use Rank 1 to this match. In our context, 
\[ A(IOF) \equiv R(IOF) \rightarrow \{ A(I) \equiv R(I) \cap A(O) \equiv R(O) \cap A(F) \equiv R(F) \} \]

**Plugin match**: This match occurs if A describes greater capability than that R requires.

We use Rank 0.75 to this match. In our context,
\[ A(IOF) \geq R(IOF) \rightarrow \{ A(I) \geq R(I) \cup A(O) \geq R(O) \cup A(F) \geq R(F) \} \]

**Subsume**: This match occurs if R requests greater capability than that R requires.

\[ A(IOF) \leq R(IOF) \rightarrow \{ A(I) \leq R(I) \cup A(O) \leq R(O) \cup A(F) \leq R(F) \} \]

**Intersection**: This filter reveals that not all the capabilities requested by the service matching with the advertised capabilities. We use Rank 0.25 to this match.

**Disjoint**: The requested service R does not match with the described service A according to any of the above filters.

In our context,
\[ A(IOF) \neq R(IOF) \rightarrow \{ A(I) \neq R(I) \cup A(O) \neq R(O) \cup A(F) \neq R(F) \} \]

Rank 0 is used for this match. The proposed Parameter Algorithm implemented in this paper is given below:-

**Algorithm Parameter Matchmaking Algorithm**

*Input*: Advertised_ontology A, Request R

*Output*: Degree_of_Match M

**Rank**: input_rank, output_rank, functionality_rank

*Parse A into* \( A(I_1, I_2, \ldots, I_m), A(O_1, O_2, \ldots, O_n) \) and \( A(F_1, F_2, \ldots, F_p) \)

*Parse R into* \( R(I_1, I_2, \ldots, I_i), R(O_1, O_2, \ldots, O_j) \) and \( R(F_1, F_2, \ldots, F_k) \).

\( c_1 = 0, c_2 = 0, c_3 = 0, i = 0, j = 0 \)

for each parsed \( A(I_1, I_2, \ldots, I_m), A(O_1, O_2, \ldots, O_n), A(F_1, F_2, \ldots, F_p) \) do

\( \text{if} A(I_i) \equiv R(I_k) \text{ then } c_1++; \)

\( \text{if} A(O_i) \equiv R(O_s) \text{ then } c_2++; \)

\( \text{if} A(F_i) \equiv R(F_t) \text{ then } c_3++; \)

end if

**input_rank** = compute_intermediaterank(m, c1, r)

**output_rank** = compute_intermediaterank(n, c2, s)

**functionality_rank** = compute_intermediaterank(p, c3, t)

**M** = leastof(input_rank, output_rank, functionality_rank)

**Rank** compute_intermediaterank(i, c, j)

\( \text{if} (i == c == j) \text{ then } R = 1; \)
if (i > c = j), then R = 0.75;
if (i = c < j), then R = 0.50;
if (i > c < j), then R = 0.25;
if (i != c != j), then R = 0;
}

6. Implementation

The parameter matchmaking algorithm is implemented in knowledge layer of the proposed architecture using java language in this paper. The java implemented algernon packages are used to query the ontology knowledge base. The package offers several java APIs with which various queries can be executed. The java implemented tokenizer extracts IOF from the service requester’s query by eliminating unwanted information from the query which is then compared with that of the advertised service and computes the degrees of matches. The algorithm starts with extracting IOF from the advertised service by executing appropriate algernon queries over service ontology described in PEG. The tokenizer implemented in the semantic component receives the service requester’s query and identifies IOF. The algorithm will then go through four stages as shown in Figure 2 to compute the degrees of match. The matchmaking module then performs semantic comparison of IOF of the requested service $R_{IOF}$ with that of advertised $A_{IOF}$ service individually in three stages and computes three intermediate ranks namely $I_r$, $O_r$, and $F_r$ as shown in the Fig 2. All the intermediate ranks are combined together in aggregate module and least rank is considered as the final rank.

We created background service ontology by considering different arithmetic operations and their respective parameters modeled as concepts. The domain ontology is used by the discovery module for inferring similarity between requested and advertised parameters.

The table 1 lists several services advertised and their capabilities in terms of IOF. These are represented semantically using OWLS editor. The OWLS editor provides higher level representation of web services using its Inputs, Outputs, Precondition and Effects (IOPE). We consider only Inputs and Outputs eliminating Precondition and Effect as they are not sufficiently standardized to be considered for matchmaking process.

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Functionalities</th>
<th>Inputs - datatype</th>
<th>Output - datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>add, sub</td>
<td>int, int</td>
<td>float</td>
</tr>
<tr>
<td>2</td>
<td>add, sub</td>
<td>int, float</td>
<td>int</td>
</tr>
<tr>
<td>3</td>
<td>sum, multiply</td>
<td>float, double</td>
<td>double</td>
</tr>
<tr>
<td>4</td>
<td>square, factorial</td>
<td>int, double</td>
<td>double</td>
</tr>
<tr>
<td>5</td>
<td>sum, multiply, Divide</td>
<td>float, float</td>
<td>Float</td>
</tr>
<tr>
<td>6</td>
<td>Squareroot</td>
<td>float, int</td>
<td>int</td>
</tr>
</tbody>
</table>

The table 2 shows a sample service request description. The Parameter Matchmaking Algorithm extracts IOF from the requested service description. Similarly, appropriate Algernon query is used to retrieve IOF from the advertised OWLS service descriptions.

<table>
<thead>
<tr>
<th>Requested Service Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service.Name</td>
<td>Mathematical Service</td>
</tr>
<tr>
<td>Service.InputsNumber</td>
<td>2</td>
</tr>
<tr>
<td>Service.Inputtype</td>
<td>float, int</td>
</tr>
<tr>
<td>Service.Output</td>
<td>1</td>
</tr>
<tr>
<td>Service.Outputtype</td>
<td>int</td>
</tr>
<tr>
<td>Service.FunctionalityNumber</td>
<td>2</td>
</tr>
<tr>
<td>Service.Functionality</td>
<td>Add, Sub</td>
</tr>
</tbody>
</table>

The algorithm starts comparing the parameters of the advertised with that of requested one by one and obtains various degrees of match. As discussed earlier, to infer the semantic information between the requested and advertised parameter, the algorithm refers background service ontology which is created using protégé ontology editor as shown in figure 3. Here, various mathematical functionalities, inputs and outputs parameter types have been modeled. As shown in figure 3, the datatype “int” is modeled as a subconcept to “float”, which means that type “int” represent more specific than that of “float”.

To explain the semantic comparison, we take the service request description given in the table 2 as an example. The requested inputs are float and int whereas...
the first advertised service’s inputs are int, int. In this case, one of the input parameter exactly matches while other one does not. Now, the algorithm refers the background ontology to determine the similarity between the other parameter “float” and “int”. In our ontology, the “int” type is modeled as sub concept of “float”. This information can be obtained using suitable Algernon query. It infers that the datatype “float” can be used in place of “int” type but not vice versa. It means that the advertised input parameters have got less capability than what is requested and they do not match exactly. This match is referred to as “subsume” as defined in the earlier section.

Similarly, the requested and advertised outputs do not match exactly and the relation between them is determined after referring the background ontology and is said to be as “plugin”. However, the functionalities requested and advertised matches “exactly”. The algorithm now combines the rank of respective matches obtained while comparing IOF and considers the match that has got least rank as the overall rank of the service with respect to the requested service. Hence, in this case, the match would be “subsume” as this is the match that has less rank value than plugin and exact. However, this is not a best match and hence the algorithm compares the requested parameters with the next advertised parameters till the best match is found. The advertised inputs of the second service are int and float. In this case, both the parameters match exactly and this match referred as “exact” match. Similarly, their outputs and functionalities also match exactly. Hence, this service is the best service when compared with that of the previous case and can satisfy the request. The algorithm now stops and returns the advertised service information to the requester to enable service access. Once suitable resource is obtained, jobs can be submitted to that resource using specific interface or through Gridbus Broker. [14]

6.2. Semantic Grid Portal

A Grid portal that consists of several portlets to provide required user interface for semantic description and discovery of services is developed. It provides necessary interface for the service providers to register their grid service and to describe it semantically. It also provides interface for the service requesters to submit their queries and to perform matchmaking of services.

The Service Oriented Architecture model of the proposed architecture for semantic grid service is shown in Figure 4.

Figure 4: Service Oriented Architecture of Semantic Grid using PEG

Figure 5a: Sequence diagram of service provider

Figure 5b: Sequence diagram of service requester
The sequence diagram for the service requester and the provider are shown in the Figure 5

7. Conclusion

In this research paper, we extended the capability of Globus Toolkit 4.0 by integrating Protégé ontology editor in it. This feature facilitates the Grid Service Providers to describe their services semantically through OWLS editor. The semantic description of services enables semantic discovery of services. A matchmaking algorithm is proposed that performs semantic matchmaking of services on the basis of IOF parameters. The user interface for semantic description and retrieval is developed as a portal enabling the user to interact easily with the grid environment. Several Grid service have been implemented and described semantically using PEG. The proposed Architecture using Parameter Matchmaking Algorithm can be applied for any specific applications enabling the users to access grid comfortably.

Acknowledgements

The Authors express their sincere thanks to Defence Research and Development Organization, New Delhi, India for financial support towards this research work.

References