Cloud Service Selection from Earth Science Domain

Lahiru S. Gallege
Department of Computer and Information Science
(IUPUI) Indianapolis, USA

Aboli J. Phadke
Department of Computer and Information Science
(IUPUI) Indianapolis, USA

Rajeev R. Raje
Department of Computer and Information Science
(IUPUI) Indianapolis, USA

Meghna B. Sebens
Department of Earth Sciences
(IUPUI) Indianapolis, USA

ABSTRACT
The promise of the cloud computing paradigm is to deliver computing as a utility available to anyone having an access to the Internet. Users can request on-demand software services hosted inside clouds. For a given application, many similar services that are developed independently will be hosted in a cloud. Hence, automatically selecting an appropriate service from these available choices to fulfill a particular requirement is a challenge. This selection function could be made available as a feature of a cloud-based middleware. The prevalent cloud related service selection methods employ simple attribute-based matching which may not yield the most relevant alternatives for complex domains such as Earth Sciences. proURDS [1] is a hierarchical, proactive discovery and selection service that provides multi-level matching which is more comprehensive than the typical attribute-based matching. This paper indicates a case study from the domain of Earth Sciences in which the proURDS is used to select relevant services from the available choices thereby, reducing the complexity involved in choosing a subset of services from a large space made up of service permutations.

General Terms
Service Discovery and Selection, Service Matching Operators

Keywords
Service Selection, Cloud Computing, Case Study, Earth Science Domain

1. INTRODUCTION
Cloud Computing (CC) is an attractive paradigm for designing, hosting, consuming, and delivering Internet-based services. It promises to achieve the long-awaited goal of making computing as a utility. It succeeded mainly due to the reasons of economy, ease of creation and use, flexibility, and scalability. Software realizations of CC-based applications would be achieved as coalitions of independently created services that are deployed in clouds, public and/or private. Hence, selecting appropriate cloud-based services is a critical step in composing CC-based applications. Consider a typical environmental monitoring system which can be created as an ensemble of many independently developed services. For example, when the team of Earth Scientists searches for Precipitation, Land Cover, Water Flow, Water Quality, and Weather Forecast services, multiple instances for each type of these services (e.g., deployed by USGS [2], USDA [3], NASA [4], and NOAA [5]) may be available that the team can choose from. These instances might be hosted in public or private clouds along with the necessary data sets.

Due to the large number of such available individual services, their possible permutations, and the associated inherent complexity, this task of discovery and selection of relevant service instances is a highly time consuming and error prone, especially if the team has a specific bias or is not very familiar with a particular model of service. Also, a specific service may not be able to easily couple with a particular other service, if these services operate at different time and spatial scales. These factors will further increase the complexity of the selection process. Hence, the discovery and selection of appropriate services from the available ones in clouds need to consider many dimensions such as, the underlying algorithmic and technological techniques used, the nature and types of the inputs, the Quality of Service (QoS) associated with the results, the ability to handle concurrent requests, and the cost of using the services. The prevalent CC-based service selection methods use simplistic matching semantics that use a limited set of attributes. Such an approach is not suitable in many complex applications from a variety of scientific domains including Earth Sciences.

The proURDS (pro UniFrame Resource Discovery Service) [1] is a proactive and hierarchical discovery service that uses the concepts of multi-level matching (MLM). The MLM is more comprehensive than the typical attribute-based approaches taken by other prevalent discovery services. The proURDS has been extensively experimented with in the general domain of service-oriented systems [1] and found to be performing better than other approaches while selecting the relevant services.

This paper describes the application of the proURDS to the domain of cloud-based services and its behavior in the context of a case study from the domain of Earth Sciences. The task of applying the principles of the proURDS to this case study is far from trivial due to: a) the inherent complexity of the Earth Sciences domain (e.g., the number of available services and their peculiarities such as nature of algorithms and data sets used), b) the unavailability of multi-level specifications for these services, and c) the continuous need for the involvement of an expert form that domain to decide the matching semantics and to assess the quality of the results (i.e., number of services) returned. This paper successfully addresses these challenges in the context of a specific case study which is discussed in Section 4.

2. RELATED WORK
Service selection and matching are integral parts of Service Discovery. Although, there have been many attempts to design discovery services in the context of service-oriented systems, there are only a few efforts that aim to discover

cloud-based services. For the sake of brevity, only the efforts from the domain of CC are discussed in this section.

The term Cloud Service Discovery System (CSDS) was introduced by the work proposed in [6]. The CSDS helps the users find the relevant services of interest and the cloud ontology consists of taxonomy of concepts of different cloud services. The CSDS is realized by building an agent-based discovery system that consults ontology to retrieve information (e.g., similarities of attributes of services) about services. The CSRA performs reasoning to find the similarity between services and rating of the services.

[7] provides architecture for the cloud services along with algorithms for measuring the performance. The main aim of [7] is to perform the service selection with adaptive performances and minimum cost. The service selection algorithm used in [7] is two-step based. The first step is the selection of the available service (basic keyword search) and the second step is the optimized service selection by using maximized gains and minimized cost of selection.

The work proposed by [8] highlights the benefits of CC and describes the cloud service discovery as being either a) keyword search, b) provider search, and c) service interface information. The advanced search options in [8] include search by service providers, technology platform and other meta-data information.

The Web Service Level Agreement Language (WSLA) and the associated framework [9] are also capable of addressing the service selection problem, however, within the WS service interface restrictions. SLA@SOI [10] describes the Open Cloud Computing Interface as an emerging standard that can be used to integrate different SLA management layers to control the life-cycle of the Cloud Services. Services can discover and interoperate by using the Open Cloud Computing Interface API and provide hybrid services. This approach does not include the service semantics and QoS information during the service selection.

Although, a few of these approaches use limited semantic techniques, others use the conventional approach of attribute-based matching. Such a simplistic view is not adequate to identify the most relevant services for complex CC-based applications.

3. SELECTION AND MATCHING

As indicated earlier, the proURDS is a hierarchical and proactive discovery service. As shown in Figure 1, the main components of proURDS are the Internet Component Broker (ICB), Headhunters (HHs), Active Registries (ARs) and Knowledge Base (KB). The ICB is similar to the middleware broker in CORBA and handles authentication and authorization, decodes, directs and routes user queries and presents the matching results back to the user.

The Headhunters (HHs) provide the functionalities of the service selection. They proactively collect multi-level specifications (described shortly) of services and also perform the multi-level matching (MLM).

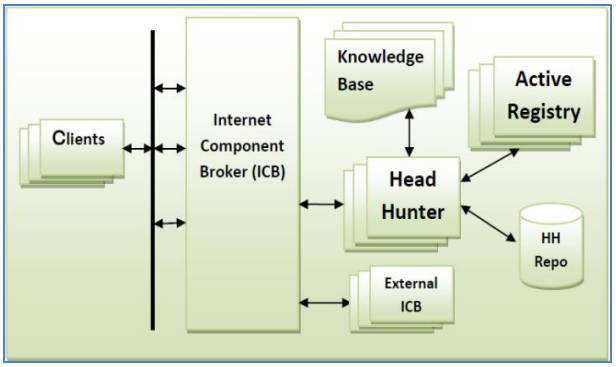

Fig 1: Basic proURDS Architecture

HHs can be homogenous or heterogeneous and can also be general purpose or specific. HHs maintain the specifications of services in associated meta-repositories. Active Registries (ARs) are the entry points for the services in proURDS. ARs communicate with associated HHs (based on access policies) on a routine basis and provide specifications to HHs. The Knowledge Base (KB) contains the necessary information to decode a query and perform associated multi-level matching. The KB is assumed to be created by experts. It consists of a problem space, and a solution space, which contains various models including the configuration knowledge to provide solutions for a family of services [11].

3.1 Multi-level Specification of a Cloud Service

Multi-level specifications (or contracts) and associated multi-level matching for software services have been suggested in [1], [12] and [13] and used by the proURDS. This multi-level specification contains levels such as syntactic, semantic, QoS and synchronization. Such a comprehensive specification serves two purposes: a) it provides a separation of concerns while designing services, and b) it enables multi-level matching that is lot more comprehensive than a single dimensional matching based on attributes. An example of a partial multi-level specification (in XML) for a Land Cover Data Service (from the domain of Earth Sciences) is indicated in Figure 2. This partial specification shows six levels: a) General b) Syntactic, c) Semantic, d) Synchronization e) QoS and f) Auxiliary.

3.2 Multi-level Matching

The multi-level matching (MLM) supported by the HHs of the proURDS uses five different levels – type, syntax, semantics, synchronization and QoS. Hence, the related section of the KB is also organized into five levels, each corresponding to the level of matching.

```
<?xml version="1.0" encoding="UTF-8"?>
<CSSContract name="USDALandCoverDataService100" type="LandCoverDataService">
  <ComponentAttributes>
    <property name="DomainName" value="EarthScience"/>
      <property name="location"  value="http://www.ers.usda.gov/Data/MajorLandUses/"/>
  </ComponentAttributes>
  <ComputationalAttributes>
      <InherentAttributes>
     <property name="license" value="gov"/>
      </InherentAttributes>
      <FunctionalAttributes>
        <SyntaxAttributes>
          <ContractAttributes>
                <Contract>
                 <property name="methodName" value="getLangUsageData" / >
                  <property name="param1"    name="location"  type="state"/ >
                  <property name="param2"    name="resolution"  type="all"/ >
                    <property name="returnType"  type="file" format="exel"/>
                </Contract>
              </ContractAttributes>
        </SyntaxAttributes>
        <SemanticAttributes>
          <PreConditon>
              <property variable="validLocation(location)" type="bool" value="true" />
           </PreConditon>
            <PostCondition>
              <property name="standardResultsFormat" value="bool" value="true"/>
            </PostCondition>
            <Invariant>
              <property name="uniformDataTimeValue" value="bool" value="true"/>
            </Invariant>
        </SemanticAttributes>
        <property name="complexity" value="O(n)"/>
      </FunctionalAttributes>
  </ComputationalAttributes>
  <QOSAttributes>
      <property name="timeFrame" scale="year(2002-2010)" value="2010"/>
      <property name="sorted" scale="bool" value="true"/>
      <property name="cost"  value="free"/>
  </QOSAttributes>
  <SynchronizationAttributes>
      <property name="MutualExclusion" implementation="websession"/>
  </SynchronizationAttributes>
  <AuxillaryAttributes>
      <property name="mobility" value="no"/>
  </AuxillaryAttributes>
</CSSContract >
```

Fig 2: Multi-level contract of a Land Cover Data Service

For example, for the type and the syntax levels, the KB contains information about the structure of types and their synonyms, inheritance hierarchy (if applicable), type compatibility, the number and order of arguments, and the return values. Similarly, for the semantic level, the KB contains a simplified ontology indicating the key terms and their relations that are used in defining pre-, post-conditions, and invariants for various services in a particular domain. The KB corresponding to the QoS level contains appropriate QoS parameters for a domain and their quantification metrics.

The semantics of the associated operators at each level (as defined in [1] and [12]) are follows. At Type level - Synonym (Exact), inheritance (Relaxed), Coercion (Relaxed); At Syntax level - Synonym (Exact), Inheritance (Relaxed), Coercion (Relaxed), Default Parameters (Relaxed), Parameter Order (Relaxed); at Semantics level - Equivalence (Exact), Implication (Relaxed), Reverse Implication (Relaxed); At Synchronization level - Compatibility; And at QoS level – Comparability. Also, as indicated in [1] each matching operator has two versions: exact and relaxed. proURDS has been extensively experimented with in the general domain of service-oriented systems [1] and found to be performing better than other approaches while selecting the relevant services. Section 4 below describes its applicability in the context of cloud-based services for a specific Earth Science case study.

4. A CASE STUDY - EARTH SCIENCES

The domain of Earth Sciences frequently involves handling of the environmental preservation activities. In such situations, teams of Earth Scientists need to perform the cause-effect analyses to conclude about the health of certain ecological systems. These analyses are achieved by the creation of distributed software systems that are composed from a variety of individual services. At present, such research teams mostly depend on human intervention to make ad-hoc choices about relevant services. For example, an ecological monitoring system called as Emergent Environment Effects Forecasting System (EEEFS) that monitors the effects of an oil spill on a body of water may consist of different types of environmental services that are hosted in public and/or private clouds along with the necessary data sets. Figure 3 shows the types of the services needed for composing the EEEFS.

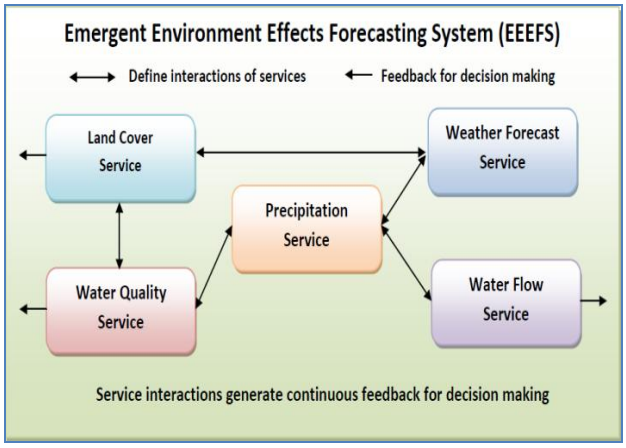

Fig 3: Architecture of the EEEFS

Selecting a proper instance of each of these types of services is based on a specific criterion that depends on the inherent nature of each type of service and also compatibility between various instances of different types. For example, the selection of an appropriate instance of the Weather Service may include considering the input/output parameter syntax details, associated semantics, and the QoS values. Due to the inherent complexity and various permutations between different instances, the EEEFS is an ideal choice to act as a case study to assess the applicability of the proURDS principles in the context of a cloud-based discovery.

To study the applicability of the proURDS and associated multi-matching principles in the context of EEEFS, the experimental infrastructure that simulated services from the categories of watershed modeling (water-flow and water quality) and spatial data modeling (land, soil, and elevation) and forecasting (weather forecasting) was created. Publically available services from online portals such as USGS [2], USDA [3], NLCD [14], SSURGO [15], and STATSGO [16] were used in the experiments. These existing services did not contain multi-level contracts and hence, their multi-level specifications were created – the main challenge in this step was to identify different instances of services (which required domain knowledge – one of the authors is an Earth Scientist) and extract the details for each level of the multi-level specification of these services. Instances of these services specifications were deployed in the experimental setup. These services were distributed randomly into the active registries of the proURDS and queries were manually written and validated against the experts' (i.e., one of the authors) domain knowledge of existing services.

```
<?xml version="1.0" encoding="UTF-8"?>
<CSSKnowladgeBase>
   <TypeRelation>
      <Synonym type="LandCoverService">
         <Type>LandCover</Type>
         <Type>LandUsage</Type>
         <Type>LandUsageService</Type>
          …
      </Synonym>
      <inheritance super="ForestCoverService" sub="LandCoverService"/>
      <inheritance super="MidWestLandCoverService" sub="USLandCoverService"/>
      <inheritance super="USLandCoverService" sub="WorldLandCoverService"/>
         …
      <coercion super="LandCoverServiceWithAllResultsByInteger"
                sub="LandCoverServiceWithAllResultsByFloat" />
         …
   </TypeRelation>

      …..
</CSSKnowladgeBase>
```

Fig 4: Partial Knowledge Base

Also, a sample KB for this domain was created in consultation with the expert. It contained domain specific information such as, service type relations - synonyms, inclusion etc. Figure 4 shows a part of this KB. As indicated earlier, the KB is consulted during the query process.

5. EXPERIMENTATION AND RESULTS

The experimental setup was made up of ten Dell machines running XP. Around 100~120 different instances from each type suitable for the EEEFS, were created to be used in the experiments. All the levels of matching were performed except the synchronization level, because the synchronization contracts for the existing Earth Science services could not be extracted due to the unavailability of their source code and most of the services use the default Web session synchronization technique.

Also, the exact and relaxed matching semantics at each of the four levels was included in the experiments (Section 3.2). Multi-Level Queries (MLQ) were issued to find the most appropriate services out of these instances. The MLQs used in the following experiments were a subset of ML-service specifications and were expressed in XML.

5.1 Query Evaluation

The first set of experiments compared the quality of the results returned by the proURDS prototype. The quality was measured as the number of relevant services returned for a particular query along with the usual metrics of precision and recall – precision is defined as the number of relevant services retrieved by a query divided by the total number of services retrieved by that query, and recall is defined as the number of relevant services retrieved by a query divided by the total number of existing relevant services (which should have been retrieved).These results were manually inspected for their relevance.

An example of such a MLQ for a Land Cover Service is shown in Figure 5. As seen from Figure 5, each query is associated with a unique ID. The query configuration level indicates how many levels of the multi-level specification should be used in the process of matching. For example, in Figure 5 this attribute is set to "3", indicating that the matching should take place at levels 0, 1, 2 and 3. Also, relaxed matching semantics is desired in this query by setting specific relaxed matching attributes to "true", as indicated in Figure 5.

```
<MCSSQuery id="21">
  <QueryConfig level="3">
    <TypeRelaxed synonyms="true" inheritance="true" coercion="true">
       True
    </TypeRelaxed>
    <SyntaxRelaxed synonyms="true" inheritance="true" coercion="true" default="true"
                   order="true">true</SyntaxRelaxed>
    <SemanticsRelaxed impl="true" revImpl="false" eq="false">true</SemanticsRelaxed>
    <QoSRelaxed compatibility="true" >true</ QoSRelaxed>
  </QueryConfig>
  <ComponentType>LandCoverData</ComponentType>
  <SyntaxAttributes>
    <ContractAttributes>
      <Contract>
        <property name="methodName" value="GetLandCoverData"/>
        <property name="parameter" value="zipcode" type="string" key="default"/>
        <property name="returnType"  type="file" format="exel" />
      </Contract>
    </ContractAttributes>
  </SyntaxAttributes>
  <SemanticAttributes>
      <PreConditon>
        <property variable="postalcode" type="string" cond="greater" value="0" />
      </PreConditon>
      <PostConditon>
        <property name="standardResultsFormat" value="bool" value="true"/>
      </PostConditon>
      <Invariant>
        <property name="uniformDataTimeValue" value="bool" value="true"/>
      </ Invariant>
  </SemanticAttributes>
  <QoSAttributes>
      < QoSAttribute>
        <property variable="timeFrame" type="year" value="2010" />
        <property variable="sorted" type="bool" value="true" />
        <property name="cost"  value="free"/>
      </ QoSAttribute>
  </QoSAttributes>
</ MCSSQuery>
```

Fig 5: Sample Query: All level relaxed matching

Table 1 indicates the comparison of results for various queries. The exact matching at all levels did not yield any results for the Land Cover Service. However, relaxing the semantics of the operators at each level resulted in a couple of matching instances for the Land Cover Service. Table 2 indicates the relaxed selection criteria, specified by the domain expert, used in the experiments.

Table 1. Land Cover Service Query Results

Query Level	Type	Syntax	Semantic	QoS
Exact Matching	51	22	6	0
Relaxed Matching	65	25	8	2

Many more queries were executed in different experiments with both the exact and relaxed matching semantics. Figure 6 shows the results of a few of these experiments. Here, for various types of queries the number of matching services after each level of matching and with exact and relaxed semantics is shown. As seen from the Figure 6, there is an increase in the number of matching services in the case of the relaxed semantics as opposed to the exact matching semantics. This is as expected due to the inherent weak nature of the relaxed matching operators.

Table 2. EEEFS Relaxed Matching Criteria

Service Type	Possible Relaxed Matching Criterion
Weather	(1) Minimize the distance from desired latitude-longitude or any other location indicator, (2) minimize cost
Precipitation	(1) Minimize the time duration overlap, (2) minimize the cost
Water Flow	(1) Minimize the distance from desired latitude-longitude or any other location indicator, (2) maximize the overlapping time duration with respect to the desired time duration, (3) minimize cost
Water Quality	(1) Maximize the overlapping water quality parameters with respect to the desired water quality variables.
Land Cover	(1) Maximize the overlap time of the map published, (2) minimize the distance between the grid size and the desirable grid size.

Tables 3 and 4 indicate the precision and recall values for these queries using the exact and relaxed matching semantics respectively. As seen from these tables, it is evident that relaxed matching resulted in better precision and recall values.

Additional experiments were conducted to test the performance, as indicated by the time required to carry out the matching operations of the proURDS prototype. The Matching Time (Tm) was used as a metric in this set of experiments. Tm is defined as the time taken by the proURDS Headhunter (HH) to perform the MLM depending on its capabilities. If a HH performs matching at all the five levels, then Tm is the sum of matching times observed at each level. Tq is summation of Tm and the time required for propagating a query to a particular HH and bringing the results back, and thus, Tq indicates the end-to-end response time for a query.

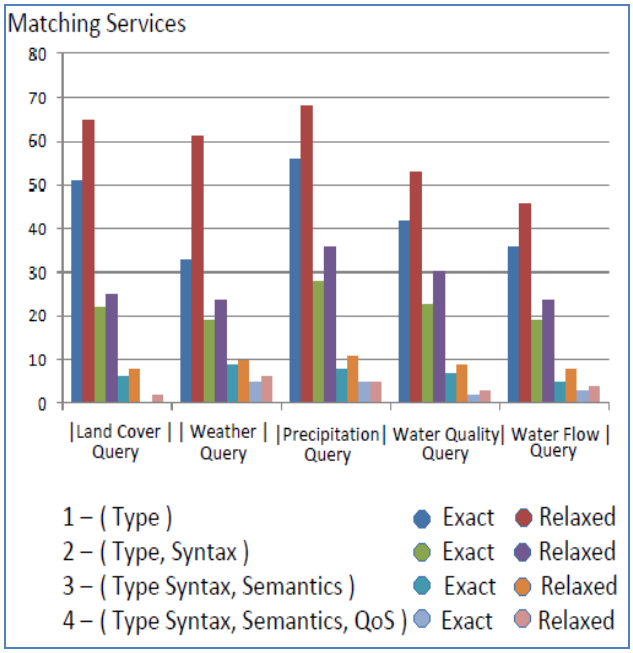

Fig 6: Comparison of Quality of Result (Exact / Relaxed)

Table 3. Exacts Matching Results

Query #	Total number of Relevant Services	Number of Returned Services	Number of Relevant Services in the Result	Precision %	Recall %
1. Land Cover	2	0	0	0	0
2. Weather	6	5	4	80	66
3. Precipitation	5	5	3	60	60
4.Water Quality	3	2	1	50	33
5. Water Flow	4	3	2	66	50

Table 4. Relaxed Matching Criteria

Query #	Total number of Relevant Services	Number of Returned Services	Number of Relevant Services in the Results	Precision %	Recall %
1. Land Cover	2	2	2	100	100
2. Weather	6	6	6	100	100
3. Precipitation	5	5	5	100	100
4.Water Quality	3	3	3	100	100
5. Water Flow	4	4	3	75	75

5.2 Performance Evaluation

Figure 7 shows the matching times required at each level for the both the semantics (exact/relaxed) of five types of EEEFS queries. As expected, each level of matching increases the response time. It is evident, from Figure 7, that the increase in the time required for the semantic matching is substantially more than the other levels, as it involves the use of a predicate proving with theorem prover [17] to establish necessary relation between the semantic part of the query and the semantic specifications of the available Earth Sciences services.

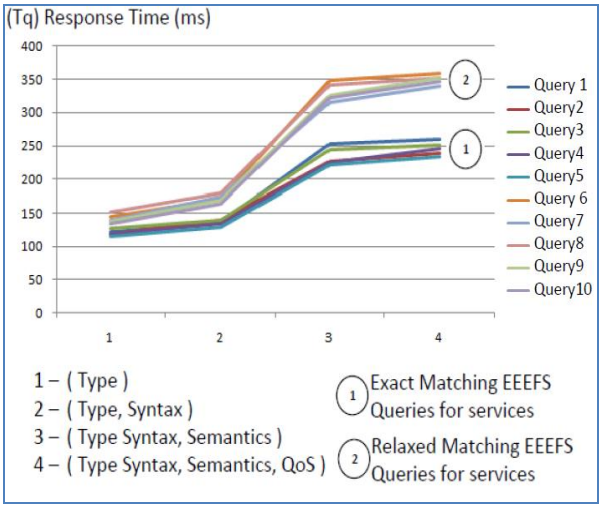

Fig 7: Individual Matching Times

Also it can be seen that among the two groups of queries, there is a tendency to increase response time in relaxed matching due to weak matching semantics and associated KB inferences. Similarly, it is evident that Tm increases as a function of number of services.

6. CONCLUSION AND FUTURE WORK

Selecting appropriate services from a set of available ones over deployed in a cloud is a crucial, laborious, and possibly error-prone step. An automated and more complete approach (than the prevalent ones) is needed to discover and select such relevant services. The proURDS, a previous work of authors, is one such hierarchical discovery system that uses the principles of multi-level specifications and associated matching. This paper has empirically validated the applicability of the proURDS in the context of cloud-based services (available from public sources) for a case study from the Earth Sciences domain.

The results, described here, indicate that proURDS returns relevant cloud services (i.e., better quality) as a result of the multi-level matching semantics at the cost of an increased response time. The future work includes further experimentation with the proURDS, research on the effects of cloud service distribution topology on the matching process, creation of additional levels (e.g., legal and cost) for matching, and an investigation of multi-level matching in the context of uncertainty and incomplete service specifications.

7. REFERENCES

[1] Gallege, L., Pradhan, K., and Raje, R., "Experiments with a Multi-level Discovery System", In Proceedings of the international Conference in Computing (ICC 2010), New Delhi, December, 2010.

[2] United Stated Geological Survey (USGS), http://eros.usgs.gov

[3] United Stated Department of Agriculture (USDA), http://www.ers.usda.gov/Data/MajorLandUses/

[4] National Aeronautics and Space Administration, (NASA) http://nasadaacs.eos.nasa.gov/

[5] National Oceanic and Atmospheric Administration (NOAA)http://www.ngs.noaa.gov/products_services.htm

[6] Han, T., and Sim, K., "An Ontology-enhanced Cloud Service Discovery System" http://www.iaeng.org/ publication/IMECS2010/ IMECS2010_pp644-649.pdf

[7] Zeng, W., Zhao, Y., and Zeng, J., "Cloud service and service selection algorithm research", In Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation, Shanghai, China, 2009.

[8] Infosys Cloud Computing White Paper http://www.infosys.com/cloud-computing/white papers/Documents/service-exchange-cloud.pdf

[9] Ludwig, H., Keller, A., Dan, A., King, R., and Franck, R., "Web service level agreement (WSLA) language specification",IBM, 2003, http://www.research.ibm.com/ wsla/WSLASpecV1-20030128.pdf.

[10] Metsch, T., Edmonds, A., and Bayon, V., "Using Cloud Standards for Interoperability of Cloud Frameworks" http://sla-at-soi.eu/wp-content/uploads/2010/04/ RESERVOIR-SLA@SOI-interop-techReport.pdf

[11] Czarnecki, C., and Eisenecker, U. "Generative Programming", Addison-Wesley, June 2000.

[12] Raje, R., "UMM: Unified Meta-object Model for Open Distributed Systems", In Proceedings of the Fourth IEEE International Conference on Algorithms and Architecture for Parallel Processing 2000.

[13] Zaremski, A., and Wing, J., "Specification matching of software components", In Proceedings of SIGSOFT'95 Third ACM SIGSOFT Symposium on the Foundations of Software Engineering, pages 6–17, October 1995.

[14] National Land Cover Data (NLCD) http://www.usgsquads.com/prod_NLCD.htm

[15] Soil Survey Geographic Data (SSURGO) http://soils.usda.gov/survey/geography/ssurgo/

[16] United Stated General Soil Map (STATSGO) http://soils.usda.gov/survey/geography/statsgo/

[17] The Java Theorem Prover, An object-oriented modular reasoning system, http://www-ksl.stanford.edu/ software/jtp/.