XMLView: Discover Domain Specific Service Description in a UDDI Compliant Registry

Weijian Fang Luc Moreau
School of Electronics and Computer Science
University of Southampton, Southampton, SO17 1BJ, U.K.
{wf, l.moreau}@ecs.soton.ac.uk

Abstract

In recent years we have seen the wide adoption of Web services architecture in grid computing. Service discovery, as one of the critical tasks in Web services architecture, is addressed by the UDDI specification. However, UDDI does not gain as much popularity as other Web services specifications such as WSDL and SOAP. At the same time, a number of domain-specific service description models are developed to address the problem of service discovery in specific application domains of grid computing. In this paper, we argue that both UDDI and domain-specific service description models can complement each other, instead of replacing each other. To demonstrate this, we present a system called XMLView that is able to accommodate domainspecific service descriptions and present them in a UDDI compliant view. Queries expressed on the domain-specific model are also supported.

1 Introduction

The past few years have seen the wide adoption of *Web services architecture* [1] in *grid computing* [2]. Service discovery, as a critical task underpinning service invocation, service orchestration, and service monitoring, is addressed by the specification of Universal Description, Discovery and Integration (UDDI) [3] in the Web services architecture. However, UDDI has not gained as much popularity as other pillars of the Web services architecture, such as WSDL and SOAP.

At the same time, several service description models have been developed inside the grid computing community to facilitate service publication and discovery, where a service-oriented architecture, but not particularly Web services architecture, is adopted. For instance, the GLUE schema [4] is widely used in grid computing to describe grid resources and services available in virtual organizations; the myGrid project (see www.mygrid.org.uk) develops a workflow environment for bioinformatics, in which a FETA

schema [9] is used to describe services available inside the myGrid workbench, where services are not limited to Web services.

It is not surprising that *domain-specific service description models* have been designed and implemented by application developers. By domain-specific service description model, we mean a non-UDDI data model that describes services for discovery in a particular application domain. Designing a domain-specific model clearly provides more flexibility to explicitly address domain specific issues, which UDDI fails to address or does not provide a straightforward solution to. For instance, domain-specific service metadata can be defined directly as a first-class member of the service model with the metadata type as the member name, while in UDDI this has to be done by means of a non-trivial generic metadata mechanism, which allows references to some predefined technical documents.

Although UDDI presents a widely-supported service discovery model, it is however difficult to persuade application developers to migrate from domain-specific models to the UDDI model, not only because UDDI's incapability and inflexibility to address some domain-specific issues, but also because the benefits from adopting UDDI might not offset the effort spent refactoring legacy code.

We argue that both domain-specific modes and the UDDI model have their own merits: domain-specific models address domain problems better, while the UDDI model is more widely understood, has more support, and thus provides better interoperability. Therefore both models should complement each other rather than replace each other.

In our previous papers, we described a UDDI-compliant Web service registry, GRIMOIRES (see www.grimoires.org), that provides extensions such as third party annotation and service discovery by interface signature [5], as well as service lifecycle management based on the WSRF specifications [6].

In this paper, in the context of GRIMOIRES, we describe how a UDDI compliant registry is able to accommodate domain-specific service descriptions and how service dis-

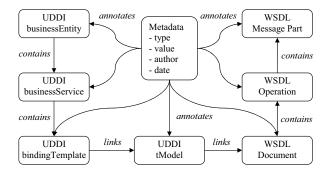


Figure 1: GRIMOIRES service description model

covery compliant with domain-specific models can be supported. The result is *XMLView*, which consists of a new web service interface in the GRIMOIRES service registry and an Eclipse-based client. When deploying GRIMOIRES in an application domain, XMLView has the capability to make GRIMOIRES' service description data model invisible to application developers. Application developers still stick to their own model to publish and discover services. Furthermore, using XMLView helps present their services in a view compliant with UDDI and leverage GRIMOIRES' registry functionality such as security, access control, and third-party annotation for free.

The rest of the paper is organized as follows. Section 2 introduces various service description models. Section 3 and 4 respectively present the two key technologies underpinning XMLView, i.e, translation between domain-specific models and the GRIMOIRES model, and template-based service discovery. Section 5 discusses the performance of XMLView. Section 6 discusses related work and Section 7 concludes the paper and outlines areas for future research.

2 Service Description Models

In this section, we introduce the data model used by GRIMOIRES to describe services. We also introduce two domain-specific service description models developed in the grid computing community: FETA and GLUE. By doing so, we show that GRIMOIRES's service description model is generic enough to express information delivered in FETA and GLUE models, i.e., both GLUE and FETA models can be mapped to the GRIMOIRES model without loss of information.

2.1 GRIMOIRES

GRIMOIRES' service description model is UDDI compliant, as seen in Figure 1. UDDI data model is composed from a limited set of high-level data constructs: *businessEntity* represents service providers, which contains *businessService* representing services; businessService further contains *bindingTemplate* representing the concrete binding of a service; and *tModel* (technical model) defines knowledge

referable from service descriptions, such as a category system or a technical interface of a service.

UDDI allows businessEntity, businessService and bindingTemplate to contain *keyed references* into tModel instances, which serves as a generic metadata annotation mechanism.

Although UDDI uses tModel to represent a service technical interface, e.g., a WSDL document, the WSDL document itself is not part of the UDDI data model. Thus UDDI provides no means to discover a service according to its interface signature, for instance, to find a service that supports a certain operation, or to find a service that is able to process a certain data type. GRIMOIRES' data model extends UDDI's to address such limitations.

First, the WSDL data model is added as part of the GRI-MOIRES' data model. The motivation is to allow the content of a WSDL document to be attached to a UDDI service description, and to allow services to be discovered based on the information contained in WSDL.

Second, a new generic metadata model is introduced. Metadata can be attached not only to UDDI constructs such as businessEntity, businessService and bindingTemplate, but also to WSDL elements such as WSDL operation and WSDL message part. A metadata element contains a metadata type that specifies the semantics of such an annotation, a metadata value, and some provenance information such as the metadata author and the annotation date. A metadata value can be a simple string, a URI for some shared ontology concept, an XML document, or even a graph of RDF triples [8].

Compared with UDDI's default metadata mechanism, GRIMOIRES's metadata model has two advantages. First, XML or RDF styled metadata value can be used to represent sophisticated and structured information as metadata, while UDDI only supports key value pairs as metadata. Second, with provenance information embedded in metadata, GRI-MOIRES allows service providers, service consumers and even third parties to attach metadata to service descriptions. The rationale behind such a design is that people other than service providers can also supply knowledge about or insight into published services, which might not be foreseen by service providers, such as a special usage of the service or user-rated quality of service.

2.2 FETA

The FETA service description model [9] was conceived by the myGrid project to describe services available in the myGrid workbench. In the FETA model, a *serviceDescription* element describes a service; it contains information about the service such as its name, description, *locationURL* (the service endpoint), *interfaceWSDL* (the service WSDL URL), and *serviceType* (the type of the service). The serviceDescription element also contains an *organization*

element that describes the service provider, and an *operations* element that describes the set of operations that this service exposes. Each operation description contains information such as *operationMethod* (an ontology concept denoting the algorithm used in the operation implementation). For each operation description, input and output *parameters* are described, which contain information such as parameter's *semanticType* (an ontology concept denoting the semantic type of the parameter).

Obviously FETA serviceDescription can be mapped to businessService in the GRIMOIRES model. FETA serviceDescription's locationURL and interfaceWSDL can be expressed by bindingTemplate in the GRIMOIRES model. FETA serviceDescription's serviceType can be mapped to metadata attached to businessService in the GRIMOIRES model. FETA serviceDescription's organization can be mapped to businessEntity in the GRIMOIRES model.

FETA serviceDescription's operations and parameters constitute the technical interface of the service, and there is no difficulty for them to be expressed using WSDL operations and message parts in the GRIMOIRES model. Furthermore, the information associated with FETA operation and parameter, such as operationMethod and parameter's semanticType, can be mapped to metadata attached to WSDL operation and WSDL message part in the GRIMOIRES model.

2.3 GLUE

The GLUE schema [4] is widely used in grid computing to describe grid resources and services available in virtual organizations. Although initially conceived as a resource description model, GLUE can also be applied to a service-oriented environment. We show that the GRIMOIRES model can also be used to fully express the information delivered in GLUE.

The core entities of GLUE include Service and Site. The latter can be considered as a description of a service provider. In addition to Service, Site contains other child elements such as Cluster (describing the computation infrastructure provisioned by the Site), Computing Element (describing the Site's job submission service) and Storage Element (describing the Site's data storage service).

GLUE Site and Service can be mapped to businessEntity and businessService in the GRIMOIRES model. Leveraging GRIMOIRES' support for attaching XML representation as metadata, other entities in the GLUE model such as Cluster, Computing Element and Storage Element can be attached to businessEntity as is in the GRIMOIRES model.

3 Translation of Service Description Models

In order to publish domain-specific service descriptions into the GRIMOIRES registry, we can choose between two strategies:

One is to introduce domain-specific models into GRI-MOIRES, and to partition the backend database for different models, with each partition only serving descriptions of a particular model.

The other is to translate domain-specific service descriptions to the GRIMOIRES model on publication and to translate back on returning as a query result.

If taken, Option 1 would support direct publication and discovery of domain-specific service descriptions with a relatively small effort. However, it would undoubtedly complicate the internal database management, which would introduce more database maintenance problems in the long run. Also, it seems non-trivial to present a service described by a domain-specific model in a view according to the GRI-MOIRES model, if required. Without such a function, users can not browse or search GRIMOIRES in a uniform way independent of service description models.

Although challenging to implement, Option 2, which translates between domain-specific models and the GRI-MOIRES model, unifies descriptions from different models in the GRIMOIRES model, and offers a solution to the interoperability between models. For these reasons, we have adopted Option 2.

In Section 2, we have already demonstrated the feasibility of mapping between domain-specific models and the GRIMOIRES model. In this section, we discuss its design and implementation. The query over domain-specific model is discussed in Section 4.

3.1 Configurability and Extensibility

Most domain-specific service description models such as GLUE and FETA can be described by an XML schema, and so does the GRIMOIRES model. XSL is widely adopted as a mechanism to translate between XML documents following different schemas. Therefore using the XSL transformation to translate between service description models reinforce the generality of our approach. We call the resulting system XMLView because it is able to handle domain-specific service descriptions that can be serialized to XML.

XMLView needs to support various domain-specific models, which may be unknown at the time of design and implementation of XMLView. Thus the *extensibility* is the key for the feasibility of XMLView: XMLView must have the capability to support new domain-specific models. Furthermore, since XMLView will be extended by GRIMOIRES users who are not familiar with GRIMOIRES internals, XMLView should also be extensible in a simple way, preferably in a declarative manner instead of a programmatic manner. Thus XMLView is designed with *configurability* deeply in mind. Figure 2 defines a named set of *translators* that specify how to translate between the FETA model and the GRIMOIRES model.

To provide a systematic way to deal with different

```
<translators elementNamespaceURI=</pre>
  "http://www.mygrid.org.uk/feta/">
  <n2qTranslator
   elementLocalname="serviceDescription"
   validate="true">
    <n2qAction type="saveBusiness"</pre>
      xsl="feta/toUDDIBusiness.xsl"/>
    <n2gAction type="saveService"
     xsl="feta/toUDDIService.xsl"/>
    <n2gAction type="saveWSDL"
      xsl="feta/toWSDL.xsl"/>
    <n2gAction type="saveMetadata"
     xsl="feta/toServiceMetadata.xsl"/>
  </n2gTranslator>
  <q2nTranslator
   grimoiresElementID="service">
    <g2nAction type="loadService"
      xsl="feta/fromUDDIService.xsl"/>
    <g2nAction type="loadBusiness"
      xsl="feta/fromUDDIBusiness.xsl"
      mountPoint="/"/>
    <g2nAction type="loadServiceMetadata"
      xsl="feta/fromServiceMetadata.xsl"
      mergePoint="/"/>
    <g2nAction
      type="loadMessagePartMetadata"
      xsl="feta/fromMsqPartMetadata.xsl"
      mergePoint="//*[local-name()=
        'parameter']"
      filter="parameterName"/>
  </g2nTranslator>
</translators>
```

Figure 2: XMLView FETA configuration

domain-specific models, XMLView introduces two artifacts in its configuration: *translator* and *action*.

A translator defines rules to transform an XML element in one model to an XML element in another model. There are two types of translators: the first translates from domain-specific model elements to the GRIMOIRES model (n2gTranslator), whereas the second performs the reverse procedure (g2nTranslator). n2gTranslator eventually saves everything into GRIMOIRES' backend database. In Figure 2, we can see an n2gTranslator is defined to recognize and translate the FETA serviceDescription element, and an g2nTranslator to translate a GRIMOIRES service element back to a FETA serviceDescription.

Rules defined in a translator are enforced by actions, which are units of functionality exposed by XMLView and reusable by different translators. Actions in a translator are invoked in the same order as they are defined. Similarly to translators, actions have two types, which we now describe.

n2gActions are used in n2gTranslators. They use the declared XSL transformation via the "xsl" attribute, to transform the submitted domain-specific elements to some GRI-MOIRES model elements, and then save into GRIMOIRES' backend database. For instance in Figure 2, saveBusiness,

saveService, saveWSDL and saveMetadata n2gActions, save information transformed from domain-specific description as UDDI businessEnity, UDDI businessSerivce, WSDL, and metadata respectively.

g2nActions are used in g2nTranslators. They load GRI-MOIRES model elements from the backend database and transform them to domain-specific elements using the declared XSL transformation. For instance in Figure 2, the loadServiceMetadata g2nAction loads metadata attached to the service and transforms it to some domain-specific element.

When a domain-specific model element is translated to the GRIMOIRES model, its contained information is represented using UDDI, WSDL and metadata models collectively. Thus when it is re-constructed, the information scattering among UDDI, WSDL and metadata models must be combined in a correct way. Hence, we introduce some directives to control this combining procedure. Currently two directives, *mount* and *merge*, are supported, assuming a result XML element is available but only contains incomplete information.

We can mount an XML element as a child to the result XML element at a specified position. The position is in the form of an XPath expression with respect to the result XML document. For instance in Figure 2, the load-Business g2nAction mounts the FETA organization element transformed from UDDI businessEntity at the position "/", i.e., as a direct child at the root of the result FETA serviceDescription.

We can also merge an XML element with the result element or a child element in the result element. The elements to be merged have the same namespace and local name, and each contains incomplete information. The merge position can also be regulated by an XPath expression. For instance in Figure 2, the loadServiceMetadata g2nAction generates a FETA serviceDescription element from GRIMOIRES service metadata, which then is merged with the result FETA serviceDescription.

We can further choose the target merge position among multiple candidates using a *filter*. For instance in Figure 2, the loadMessagePartMetadata g2nAction generates a FETA parameter element from GRIMOIRES message part metadata, which then should be merged with the FETA parameter element of the same *parameterName* in the result FETA service description. Note there are multiple parameter definitions in one FETA serviceDescription. Here, a merge point "//*[local-name()='parameter']" tells that the element to be merged is of the local name "parameter", and a filter "parameterName" says that only the parameter of the same parameterName should be merged.

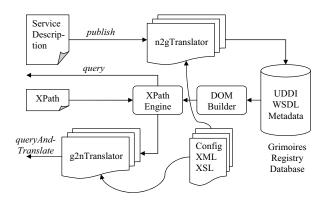


Figure 3: The architecture of XMLView

3.2 Architecture

Our previous paper [6] presents the architecture of the GRIMOIRES registry. Figure 3 gives the architecture of XM-LView. XMLView adds a new Web service interface to the GRIMOIRES registry. Within this interface, four operations are defined:

The *publish* operation takes an arbitrary XML element for publication. It is up to XMLView to decide whether to accept it, i.e., to commit the publication into GRIMOIRES, or to reject it. The latter situation probably means there is no translator registered for this particular element. Otherwise, the right translator corresponding to the to-be-published element is invoked. The interface places no constraint on the sort of domain specific service description that is eligible to be published as long as it has an XML representation.

The *delete* operation (not shown in Figure 3) takes a string key and deletes the GRIMOIRES registry entity to which the key refers. The key for instance could be a UDDI business key or a UDDI service key.

XMLView offers two query operations: *query* and *queryAndTranslate*, which are discussed below.

XMLView presents all information registered in GRI-MOIRES, whether UDDI, WSDL or metadata, as a single unified DOM document. This DOM document should be considered virtual rather than concrete: it has no backing XML document file, it is generated on demand as well as is cacheable, and caching can be made at finer grains rather than at the document level.

Both query and queryAndTranslate take an XPath expression as input, and return the result by executing the input XPath over the DOM document presented by XM-LView. The difference between these two operations is that queryAndTranslate will attempt to translate the query result to XML elements compliant with the specified domain-specific model whenever possible.

Let us examine the queryAndTranslate operation in more detail. In addition to the XPath expression, queryAndTranslate also takes as input the XML namespace for the expected domain-specific model, for instance a namespace

```
<feta:serviceDescription xmlns:feta="...">
   <feta:serviceName>
      SHound3DbWhat
   </feta:serviceName>
</feta:serviceDescription>
```

Figure 4: A FETA query template to find a "SHound3DbWhat" service

"http://www.mygrid.org.uk/feta/" indicates the query result is expected to be translated to the FETA model. This namespace prompts XMLView which translator should be used. For instance, if a UDDI businessService appears in the query result, XMLView checks whether there is a registered translator able to translate from UDDI businessService to some element in the expected domain-specific model. In fact, XMLView also checks whether this found UDDI businessService is originally from the expected domain-specific model. Only if it is from the expected domain-specific model, XMLView will perform the translation. Service descriptions in different domain-specific models usually contain quite different information, and thus sometimes, it does not make sense to translate between different domain-specific models.

Because the proper translator can be chosen according to the qualified name of the element to be translated, an XM-LView instance supports multiple domain-specific models, as well as GRIMOIRES's own model, at run time.

4 Template-based Query

To use the query or queryAndTranslate operation, an XPath expression conforming to the GRIMOIRES model must be composed. Thus on the basis of this interface, the GRIMOIRES model is still not made invisible to domain application developers.

To solve this problem, we propose a *template-based query* technique. The idea is, in order to discover some service description according to a certain domain-specific model, users need to create an incomplete description that contains known information about the service as search criteria. Note such a description could be invalid according to the model since some required information could be missing from the template. For instance, Figure 4 shows a FETA query template that finds a "SHound3DbWhat" service.

Given a query template, we can then generate an XPath expression conforming to the GRIMOIRES model, which involves two steps:

First, translate the query template in the domain-specific model to some template(s) in the GRIMOIRES data model using the method described in Section 3. Such translation is possible because the query template is expressed according to the domain-specific schema supported by the translation. For the example template given in Figure 4, the GRIMOIRES query template is shown in Figure 5.

Second, generate the XPath expression from the GRI-

```
<uddi:businessService xmlns:uddi="...">
    <uddi:name>
        SHound3DbWhat
      </uddi:name>
    </uddi:businessService>
```

Figure 5: The translated query template in the GRIMOIRES model

```
//uddi:businessService
[uddi:name='SHound3DbWhat']
```

Figure 6: The XPath expression conforming to the GRIMOIRES model

MOIRES query template. The generated XPath expression selects from the GRIMOIRES DOM document those elements having the same namespace and local name as the given GRIMOIRES query template(s), and containing all the information existing in the given template(s). For the example given in Figure 5, the generated XPath is shown in Figure 6.

Having generated the XPath expression compliant with the GRIMOIRES model, we then can use it to query over the GRIMOIRES registry. The query result can be translated back to the domain specific model, if required.

Using the template-based query technique, the GRI-MOIRES service description model is made totally invisible to domain application developers when service discovery is concerned.

To make it user-friendly, template-based query is implemented inside an Eclipse plug-in, called GRIMOIRES Browser. As a GUI to the GRIMOIRES registry, GRIMOIRES Browser has full support of the GRIMOIRES registry's functionalities including the XMLView interface.

5 Evaluation

In this section, we evaluate the performance of the XM-LView service. The general discussion of GRIMOIRES' performance can be found in our previous papers [7] and [6]. We discuss two domain-specific models for which the translations are provided in XMLView: GLUE and FETA. In each case, we present the service publication and discovery overheads. Of course, users can extend XMLView to support their very own model. In that case, the performance we report here can provide valuable information to estimate and reason about the performance with respect to that user-defined domain-specific model.

The experiments in Section 5.1 and 5.2 are running on a Windows XP computer installed with an Intel Core 2 Duo 2GHz Processor and 2GB memory. Both the test client and the GRIMOIRES registry with XMLView are on the same computer. The throughput experiment in Section 5.3 is carried on the Iridis cluster (see http://www.soton.ac.uk/iss/computing/hpc/iridis/index.html).

5.1 FETA

We take 79 FETA descriptions from the myGrid project. It takes 116 seconds to publish them sequentially into GRI-

Table 1: FETA service discovery criteria

	In the FETA model	As in the GRIMOIRES model
SN	service name	UDDI businessService name
ST	service type	metadata attached to
		UDDI businessService
OT	operation task	metadata attached to
		WSDL operation
PST	parameter semantic	metadata attached to
	type	WSDL message part

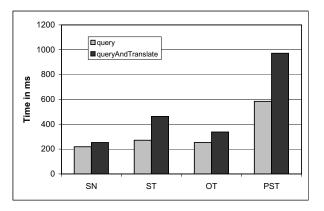


Figure 7: Service discovery for the FETA model

MOIRES through the XMLView interface, i.e., it takes an average of 1.47 second to publish a FETA description. In average, a FETA description contains 1934 bytes and 30 XML tags. The populated GRIMOIRES registry is then used to evaluate the service discovery performance.

In this experiment, it is relatively expensive to publish a FETA description because all 79 sample FETA descriptions, having been translated to UDDI businessServices, belong to the same UDDI businessEntity. Before adding a businessService into a businessEntity, the GRIMOIRES registry needs to check whether such a businessService already exists in the businessEntity. If so, the old businessService would be deleted. Thus adding a businessService into an existing businessEntity incurs an overhead relevant to the size of the businessEntity. So the relatively expensive publication overhead that we observe in this experiment is rather associated with these particular FETA description samples than a general characteristic about XMLView's publication performance.

Four service discovery cases with different query criteria are evaluated, as listed in Table 1. Their performance are shown in Figure 7. For each case, we measure the overhead of both the query operation (shown in the grey bar) and the queryAndTranslate operation (shown in the black bar). In addition to what has been done by the query operation, the queryAndTranslate operation translates matched descriptions in the GRIMOIRES model back to the FETA model.

The overhead of the query operation is decided by the complexity of the XPath expression used in the query. For

Table 2: GLUE discovery criteria

	In the GLUE model	As in the GRIMOIRES model
SiteN	(find Site by) name	UDDI businessEntity name
ServN	(find Service by) name	UDDI businessService name
CN	(find Site by) Cluster	metadata attached to
	name	UDDI businessEntity

instance, among them, PST, which finds a service that has an operation taking in a parameter of the specified semantic type, uses an XPath expression crawling over UDDI, WSDL, and metadata models, and thus not surprisingly incurs the largest overhead.

The translation overhead, measured by the difference between the two operations' overheads, is relevant to the number of FETA service descriptions being translated, and to the information complexity of the descriptions. SN returns 1 matched description, ST returns 2, OT returns 3, and PST returns 7. The average translation overhead for a single FETA service description is 53.6ms.

5.2 GLUE

We have downloaded all the GLUE descriptions available in the EGEE project (see public.eu-egee.org) from its LDAP-based information server, and transformed them into XML documents conforming to the GLUE XML schema. There were 267 GLUE Site descriptions on 12 October 2007. It takes 281 seconds to publish them into the GRIMOIRES registry through the XMLView interface. This gives an average of 1.05 second for the publication overhead per description. In average, a GLUE description contains 90,682 bytes and 456 XML tags. The populated GRIMOIRES registry is then used to evaluate the discovery performance.

Three service discovery cases with different query criteria are evaluated, as listed in Table 2. Their performance are shown in Figure 8. For each case, we measure the overhead of both the query operation (shown in the grey bar) and the queryAndTranslate operation (shown in the black bar). In SiteN and ServN, the overhead of the query operation is relatively small: 47ms for SiteN and 31ms for ServN. CN incurs a large overhead due to its complex XPath expression.

5.3 Throughput

We also conducted a throughput test using the same set of EGEE GLUE descriptions as in Section 5.2 on the Iridis cluster. During the experiment, one cluster node is dedicated to the GRIMOIRES registry, and extra 16 cluster nodes are brought up one by one as clients. Each client node starts up 8 threads, and each thread repeatedly sends random GLUE Site discovery request to the GRIMOIRES registry. A new request is sent only if the previous request has been served. Therefore ultimately there are 128 clients (in the form of threads) simultaneously accessing GRIMOIRES. We measure the total number of served requests, which is shown in Figure 9.

In Figure 9, each data point represents the number of total served requests per minute (i.e., the sum of all clients' throughput). When plotting, we take an average of five neighboring measurements. In the figure, a cross indicates the time point when a new client node is brought up.

As seen in Figure 9, after running more than 10 hours, with up to 128 clients, we do not observe obvious performance degradation when numerous clients simultaneously accessing GRIMOIRES. On 26/10/07 08:00, 10 minutes after the last client node was brought up, 3165 requests had been served in that minute. Many jumps can be seen along the curve. Although those cluster nodes are dedicated to our experiment, the cluster network is shared between our experiment and other applications running at the same time. As there is a lot of communication involved between the GRIMOIRES registry and clients, those jumps could be due to the interruption caused by the other network users.

6 Related Work

There are some other service description models in addition to those discussed above. For instance, OWL-S [10], formerly DAML-S, uses the OWL Web ontology language to describe a Web service; BioMOBY [11] defines a service as an atomic process or operation that takes a set of inputs and produces a set of outputs; intended to be a generic information management system, the ebXML registry [12] also defines a registry information model and a Web service profile that implements a Web service registry. With the help of XMLView, GRIMOIRES can be set up to accommodate OWL-s, BioMOBY and ebXML service descriptions and serve queries according to their own models.

The Globus Toolkit (see www.globus.org) provides the Monitoring and Discovery System (MDS). MDS focuses on the mechanism to disseminate and gather information on Grids rather than the information model to describe services or resources. Information is published in xml according to some schema. While the GLUE schema is used for compute information, the owners of information sources or grid resources have the freedom to define their

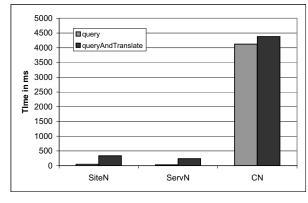


Figure 8: Service discovery for the GLUE model

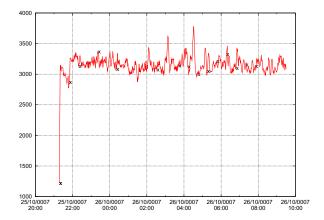


Figure 9: The throughput of XMLView

own schema, so that arbitrary xml data can be published to describe service profiles and states. As far as inquiry is concerned, MDS relies on the XPath language. Similar to MDS, XMLView allows users to define their own domain-specific model to describe services, and such a model is fully supported during publication and discovery. Different from MDS, XMLView is able to present published descriptions in a UDDI-compliant view.

Srinivasan *et al* [13] described a UDDI registry that also understands OWL-S service description. So users can directly publish OWL-S descriptions, which are then mapped to the UDDI data model by the registry. In addition to standard UDDI publication and query interfaces, a *capability* interface is provided to support ontology-based matchmaking. In comparison, XMLView takes a more generic approach in that any domain-specific service descriptions with an XML representation can be published and discovered, which include OWL-S descriptions.

The idea of using XSL to transform syntactically different but semantically equivalent documents has been explored in many places. For example, Szomszor *et al* [14] described using XSL-based type adaptors to harmonise syntactically incompatible interfaces in Web services workflows.

7 Conclusion

In this paper, we present the XMLView approach to allow a UDDI compliant service registry such as GRIMOIRES to accommodate domain specific service descriptions and to support queries according to the domain-specific model.

In this way, GRIMOIRES makes its own service description model transparent to domain applications. Using GRIMOIRES with XMLView, domain developers are able to present domain services in a standard UDDI view, and to make use of functionalities of GRIMOIRES such as third party annotation, security and access control.

Our evaluation of XMLView shows it presents acceptable performance. The throughput experiment uses the reg-

istry data from the EGEE information server, and reveals that XMLView has a stable performance under stress test. Future work will expand on this basis to evaluate XMLview's performance under more dynamic and realistic workload.

Acknowledgment

GRIMOIRES is funded by the Commissioned Software Project of OMII-UK (see www.omii.ac.uk).

References

- Web Services Architecture. The World Wide Web Consortium. http://www.w3.org/TR/ws-arch/.
- [2] Service-Oriented Science. I. Foster. Science, vol. 308, May 6, 2005.
- [3] Universal Description, Discovery and Integration (UDDI). OASIS. http://www.uddi.org.
- [4] The GLUE schema. http://www.cnaf.infn.it/ ~sergio/datatag/glue/
- [5] Simon Miles, Juri Papay, Terry Payne, Keith Decker, and Luc Moreau. Towards a protocol for the attachment of semantic descriptions to grid services. *Scientific Programming*, vol. 12, pp. 201-211, 2005.
- [6] Weijian Fang, Luc Moreau, Rachana Ananthakrishnan, Mike Wilde, and Ian Foster. Exposing UDDI service descriptions and their metadata annotations as ws-resources. In Proceedings of the 7th IEEE/ACM International Conference on Grid Computing, pages 128-135, Barcelona, Spain, September 2006.
- [7] Weijian Fang, Simon Miles, and Luc Moreau. Performance analysis of a semantics-enabled service registry. Concurrency and Computation: Practice and Experience, 2007.
- [8] WWW Consortium. RDF Primer. http://www.w3.org/TR/ rdf-primer/.
- [9] Phillip Lord, Pinar Alper, Chris Wroe, and Carole Goble. Feta: A Light-Weight Architecture for User Oriented Semantic Service Discovery. In Proceedings of Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May-June 2005. pp. 17-31, Springer-Verlag LNCS 3532 2005.
- [10] OWL-S. http://www.daml.org/services/owl-s/.
- [11] M. D. Wilkinson and M. Links. Biomoby: an open-source biological web services proposal. *Briefings In Bioinformatics*, 4(3), 2002.
- [12] ebXML Registry standards. http://www.oasis-open.org/ committees/tc home.php?wg abbrev=regrep.
- [13] N. Srinivasan, M. Paolucci, and K. Sycara. Adding OWL-S to UDDI, implementation and throughput. In First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, California, USA.
- [14] M. Szomszor, T. R. Payne, and L. Moreau. Dynamic Discovery of Composable Type Adapters for Practical Web Services Workflow. In *UK e-Science All Hands Meeting 2006*, September 2006, Nottingham.