Exposing UDDI Service Descriptions and Their Metadata Annotations as WS-Resources

Weijian Fang #1, Luc Moreau #2, Rachana Ananthakrishnan *3, Mike Wilde *4, Ian Foster *5

#*School of Electronics and Computer Science, University of Southampton*
Southampton, SO17 1BJ, U.K.
1`wf@ecs.soton.ac.uk` 2`l.moreau@ecs.soton.ac.uk`

**Mathematics and Computer Science Division, Argonne National Laboratory*
Argonne, IL 60439, U.S.A.
3`ranantha@mcs.anl.gov` 4`wilde@mcs.anl.gov` 5`foster@mcs.anl.gov`

***Abstract*— Service discovery is a critical task in service-oriented architectures. GRIMOIRES is a UDDI-compliant service registry with rich metadata annotation capabilities. In this paper, we present a WSRF-compliant extension of the GRIMOIRES architecture that exposes service descriptions and their metadata annotations as WS-Resources. By doing so, the service registry provides improved interoperability by allowing service descriptions to be accessed and managed via standard Web services operations. In particular, we can access service descriptions, manage the lifetime of service descriptions, and subscribe for notifications of change in service descriptions. The paper discusses the benefits and design choices associated with such an approach and the technical challenges in providing an implementation.**

I. Introduction

In service-oriented architectures, service discovery is a critical task underpinning service invocation, service orchestration, and service monitoring. Among the many proposals for service discovery in this context ([1], [2], [3]), the UDDI (Universal Description, Discovery, and Integration) specification [1] is the standard for Web services publication and discovery. UDDI defines both a data model to describe services and a set of interfaces to publish and discover service descriptions. UDDI suffers, however, from some limitations that hinder its widespread adoption. UDDI offers no mechanism to refer to a service interface signature (such as operations or input and output messages) and therefore is unable to discover a service according to such a signature. Moreover, UDDI provides no lifetime management and thus is unsuitable for the Grid environment, where transient services may be used extensively. Above all, UDDI lacks the capability of annotating service descriptions with structured metadata [4]; hence, it is difficult to enrich service descriptions in a way that can help future service discovery.

Having identified that metadata annotation can play a vital role in assisting service discovery, we designed and implemented GRIMOIRES, a UDDI-compliant service registry extended with a powerful metadata attachment support (`www.grimoires.org`). Adopting the UDDI standard facilitates the acceptability of the GRIMOIRES approach by the Web services community. In addition, GRIMOIRES allows both service providers and service consumers to annotate published service descriptions with metadata so as to facilitate service discovery. By exploiting GRIMOIRES' metadata attachment capability, domain-specific service annotation and service discovery by interface signature can be accomplished [4], as demonstrated in the bioinformatics e-Science project myGrid (`www.mygrid.org.uk`).

The lifecycle of a service begins with its deployment and terminates with its decommissioning. During its lifecycle, the service may vary in different ways: on the one hand, its interface can change when new operations are exposed, operation signatures are updated, or operations are removed; on the other hand, the service may be redeployed to another execution environment, which results in a change of its concrete binding. Hence, in order to remain accurate, a service description, as published in the service registry, should ideally have a lifecycle that well matches the lifecycle of the corresponding service.

The WSRF [5] and WS-Notification [6] specifications have gained increasing attention in their role of improving interoperability with typed stateful resources that have a lifecycle [7]. Specifically, stateful resources are modeled as WS-Resources, and their state can be accessed and modified in a standard way through so-called resource properties [8].

By exposing service descriptions as WS-Resources, we enhance a service registry's interoperability in different ways. Service descriptions can be published and discovered through standard uniform state-oriented operations. In particular, a general-purpose query language (such as XPath) offers more flexibility than the UDDI rigid query templates. Service descriptions annotated with lifetime information can automatically be cleaned up when their lifetime expires. Moreover, changes in the service descriptions can be delivered to subscribers using a notification mechanism. Against this background, the specific contributions of this paper are as follows:

(i) We discuss the design choices and benefits of exposing UDDI service descriptions and their metadata annotations as WS-Resources.

(ii) We analyze the technical challenges in realizing such a vision in a software architecture. In particular, we investigate the means to maintain a WSRF view of states (i.e., service descriptions and annotations) existing in the registry database.

(iii) We present several technical solutions to allow efficient publication and query through the WSRF interface, while preserving the WSRF interface's flexibility and uniformity.

(iv) We report on the prototype implementation and its performance evaluation.

The rest of the paper is organized as follows. Section II presents background information about the UDDI and WSRF specifications, as well as GRIMOIRES' metadata annotation capability. Section III explains how we expose annotated UDDI service descriptions as WS-Resources. Section IV presents the architecture of GRIMOIRES. Section V discusses the implementation of its WSRF-compliant interfaces. Section VI presents our performance evaluation of the GRIMOIRES system. Section VII discusses related work and Section VIII outlines areas for future research.

II. Background

A. UDDI

The UDDI service registry [1] is the standard for Web services discovery. Service descriptions in UDDI are composed from a limited set of high-level data constructs: Business Entity is the data model for service providers, Business Service for services themselves, Binding Template for the concrete bindings of the services, and Technical Model (tModel) for some shared knowledge such as a category system or the technical interfaces of the services.

In UDDI, a service description can reference a value as its metadata; the value is encoded in a value set represented by a tModel that thus speaks for the meaning of the value [9]. Since the description of the value set is not saved in the registry itself, UDDI acts only as a contact point to get further information and thus lacks the capability of inquiry by the content of any metadata.

The same problem arises on inquiry by a service technical interface usually defined by a WSDL document. Here, tModels are also used to record a URL to an external WSDL file. Hence UDDI does not hold interface signatures and therefore does not allow users to query a service by its interface signature. For instance, UDDI does not support discovering a service that accepts an input message of a given type. We note that a UDDI technical note [10] describes a way to discover a service based on a certain WSDL porttype.

Allowing service consumers to annotate service descriptions promotes the sharing of knowledge about services. It also allows users with expert knowledge to enrich service descriptions in flexible ways, with information that might not be available to the original publishers. However, UDDI does not specify how to attach metadata to a published service by service consumers.

B. Metadata Annotation

Since UDDI lacks support for metadata annotation and metadata-based service discovery, we designed GRIMOIRES, a UDDI-compliant registry with a rich metadata annotation capability. GRIMOIRES allows metadata to be attached to any UDDI entities. A piece of metadata is composed of a type describing the meaning of this annotation, a value and some provenance information (such as author or date). The value can be a simple string, a URI that refers to a predefined ontology concept, or a complex and structured description expressed as an RDF graph [11].

Users can attach metadata to service descriptions to provide domain-specific knowledge about the service; such knowledge can be used to identify services more precisely during service discovery. Metadata-related operations supported by GRIMOIRES include attaching, updating and deleting metadata, as well as discovering entities according to their attached metadata. The ability to publish, update and read metadata is available to both service providers and service consumers under a configurable access control mechanism [12].

With such a capability, GRIMOIRES can support service discovery based on interface signature. Indeed, users can annotate the input of a Web service operation with a semantic type, which provides a description of the input in terms of the application's semantics rather than the syntactical type encoded in the SOAP message. For instance, a nucleotide sequence can be encoded as a string in the SOAP message, but a more meaningful approach is to qualify the sequence as its semantic type. With such annotations in place, a user can discover a service that, for instance, is able to process a nucleotide sequence.

C. WSRF

WSRF introduces the concept of WS-Resource to model stateful resources. More precisely, a WS-Resource models a stateful resource that has a well-defined lifecycle and can be expressed as an XML document [7]. To this end, a resource property is seen as a piece of information defined as part of the state of a resource. Each resource property is expected to have an XML representation, referred as resource property element. All the resource property elements of a resource are logically composed in a single XML document, named the resource properties document.

The standardized operations on a WS-Resource include accessing (updating/deleting/querying) resource properties [8], immediately destroying or scheduling a later destroy of a WS-Resource [13], as well as subscribing and getting notification of the changes of a WS-Resource [6].

We emphasize that by exposing service descriptions and their annotated metadata as WS-Resources, GRIMOIRES's users can also leverage WSRF operations to publish and discover service descriptions in the registry.

III. Service Descriptions as WS-Resources

In this section, we discuss design choices in exposing service descriptions (and other registry entities) and their metadata annotations as WS-Resources that would allow users to access them in a WSRF-compliant way.

A. Resource Property

We focus here on how to expose the state stored in the registry database through the means of WS-ResourceProperties.

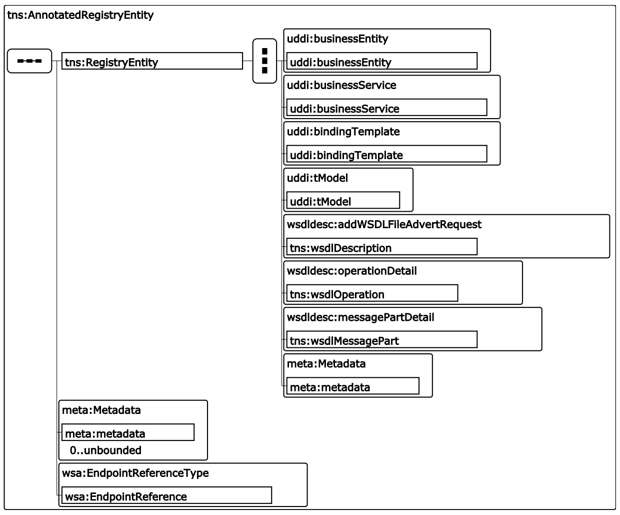

Fig. 1. AnnotatedRegistryEntity RP

Our first option is to use a single resource property (RP), to reflect a registry entity's state; this includes the registry entity itself but also its metadata annotation. We name such a representation `AnnotatedRegistryEntity`, and its XML schema is shown in Figure 1. By encapsulating in the `AnnotatedRegistryEntity` RP a representation of both the registry entity and its metadata, we build an explicit annotation relationship that specifies which metadata is associated with which registry entity. Such an annotation relationship is crucial to allow discovery of registry entities by their attached metadata.

In GRIMOIRES, metadata are entities that potentially are more volatile than the service descriptions they are associated with and can be queried and updated by dedicated operations. Hence, an alternate design would be to expose each metadata directly as a RP. This offers two benefits. First, we could make use of generic WS-ResourceProperties getters and setters to read and update GRIMOIRES metadata. Second, if some metadata needs to be updated, we would just provide the new metadata value and we would no longer need to construct a complete `AnnotatedRegistryEntity` including a service description and all its metadata.

While such a design seems appealing, the WSRF specification sets a number of constraints to its realization. GRIMOIRES metadata are (type, value) pairs with additional provenance information. In order to expose a piece of metadata as a WS-ResourceProperty, the metadata type would have to become the RP's QName (and it would also be reflected in the definition of the resource properties document that appears in the service WSDL description.). Unfortunately, we cannot adopt such a design because GRIMOIRES metadata types are provided dynamically by users and thus cannot be encoded statically in the schema of the resource properties document.

`RegistryEntity` enumerates all types of registry entities that GRIMOIRES supports to attach metadata to. Specifically, `businessEntity`, `businessService`, `bindingTemplate`, and `tModel` are defined in the UDDI data model; `wsdlDescription` contains the URL and content of a WSDL description; `wsdlOperation` is used to locate an operation defined in a WSDL description, by using its porttype namespace, porttype name, and operation name; and `wsdlMessagePart` is used to locate a message part defined in a WSDL description, by using its message namespace, message name, and message part name. Metadata can also be attached to other metadata.

Each `AnnotatedRegistryEntity` also contains a WS-Addressing endpoint reference pointing to the WS-Resource containing this `AnnotatedRegistryEntity` RP. Whenever a `AnnotatedRegistryEntity` RP is retrieved, it can be figured out where it comes from and where to get its latest version if it is suspected being obsolete.

So far, we have modeled each registry entity as a WS-Resource, with a single RP named `AnnotatedRegistryEntity`. However, the registry as a whole is itself also a stateful resource, which contains all registry entities. The XML representation of the registry as a whole is the composition of the XML representations of all its entities.

B. WSRF-*Compliant Interfaces*

We now introduce WSRF-compliant interfaces that allow users to operate on the WS-Resources described in the previous section. The registry exposes three WSRF-compliant interfaces, namely `Entry`, `Query`, and `Factory`, as depicted in Figure 2 and described below.

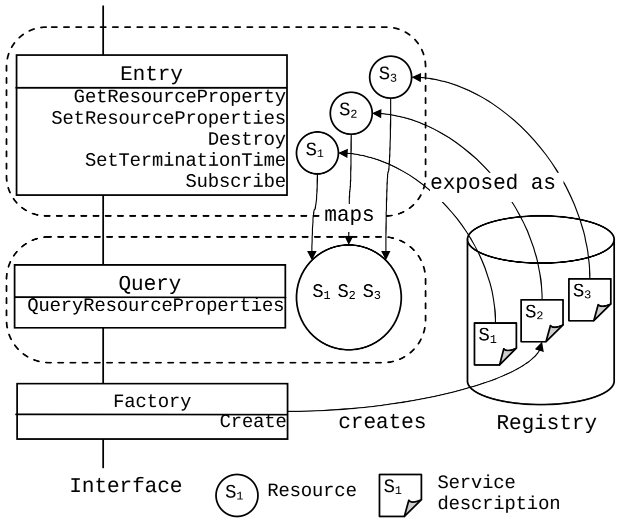

Fig. 2. GRIMOIRES WSRF compliant interfaces

`Entry` is the interface to interact with individual registry entities, such as service descriptions, exposed as stateful resources. The `Entry` interface is associated with multiple resources. Each of them corresponds to one registry entity. The `Entry` interface allows users to get and set each resource's `AnnotatedRegistryEntity` RP, as well as to delete each resource. It also allows users to subscribe to the changes of registry entities. In other words, the `Entry` interface gives users full access to individual registry entities through WSRF operations.

While the `Entry` interface provides the capability of querying over individual registry entities, the `Query` interface aims to facilitate discovery over the whole registry. The `Query` interface is associated with only one resource, which corresponds to the collection of all registry entities. Through the `Query` interface, users are presented with a view of the whole registry, which allows them to issue queries over the whole registry using a query language such as XPath in a standard WSRF operation. `Query` is a readonly interface in that it supports the query operation only over the whole registry; updates to individual registry entities must go through the `Entry` interface.

WSRF does not specify the protocol to create WS-Resources. Hence, the `Factory` interface is designed to act as a resource factory, through which new registry entities can be created as resources associated with the `Entry` interface.

Both the `Entry` resources (the resources associated with the `Entry` interface) and the `Query` resource (the sole resource associated with the `Query` interface) have `AnnotatedRegistryEntity` as their RP. However, the `AnnotatedRegistryEntity` resource property element has a cardinality of 1 in an `Entry` resource properties document and an unbounded cardinality in the `Query` resource properties document (because in the latter case the `Query` resource is seen as the collection of all `Entry` resources).

Furthermore, when the whole registry is queried over through the `Query` interface, if a registry entity is matched with given criteria, the endpoint reference to the corresponding `Entry` resource can be resolved so that users can take further actions directly on the `Entry` resource, for instance, by subscribing to its changes or by destroying it.

C. Benefits

The proposed design offers several benefits.

(i) Service descriptions can be published and discovered through standard uniform state-oriented operations. Not only does this feature enable WSRF-compliant clients to access the contents of the GRIMOIRES UDDI-compatible registry, but it also augments UDDI's capability of handling service descriptions. For instance, service descriptions can be queried over by using an XML query language, such as XPath, which is more flexible than the UDDI rigid query templates.

(ii) By following the WSRF specification, we unify different ways of accessing various data models, such as UDDI data model, metadata, and WSDL descriptions, which contribute to a complete service description in GRIMOIRES. By comparison, different APIs were previously used to publish and inquire over these different data models.

(iii) Through the mechanism of soft-state lifetime management defined in the WS-ResourceLifetime specification [13], a service description for a transient service can be annotated with a lifetime so that the registry can automatically clean up the description when its lifetime expires.

(iv) Changes in the service descriptions can be delivered promptly to subscribers through the WS-Notification mechanism. Furthermore, metadata annotation that has enriched a published service description can also be captured by subscribers through the same notification mechanism.

IV. Architecture

Figure 3 shows the architecture of the GRIMOIRES registry, which we discuss in a top-down manner.

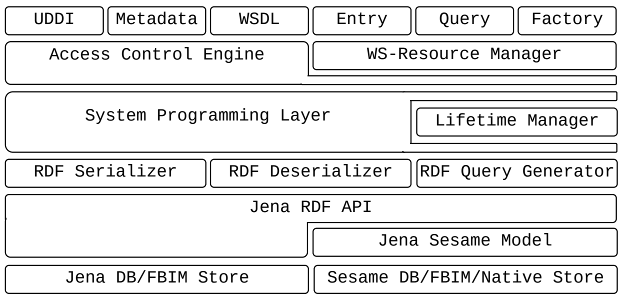

Fig. 3. GRIMOIRES Architecture

GRIMOIRES provides two types of interfaces. Non-WSRF-compliant interfaces include `UDDI` following the UDDI standard, `Metadata` for attaching metadata and querying by metadata, and `WSDL` for the publication of and query over WSDL descriptions; WSRF-compliant interfaces include `Entry`, `Query`, and `Factory` (presented in Section III-B).

The WS-Resource manager is in charge of exposing registry entities as WS-Resources (discussed in Section V).

The access control engine plays a vital role in ensuring that service descriptions as well as their attached metadata can be trusted. All requests coming through the WS interfaces reach the policy decision point of the access control engine before they—if allowed—are passed to the system programming layer. The authentication is based on the client's X.509 certificate; the access control engine itself is XACML based. (For more detail, see [12].)

The system programming layer (SPL) implements the business logic of the `UDDI`/`Metadata`/`WSDL` interfaces. The SPL calls the RDF Deserializer/Serializer to deserialize/serialize UDDI/metadata/WSDL descriptions to and from RDF triples, which reside in a triple store. The SPL also calls the RDF Query Generator to translate `UDDI`/`Metadata`/`WSDL` queries to RDF queries.

The scheduled termination time of a service description is expressed as a special metadata item attached to that service description. A service provider is able to attach this metadata to the published service, which has a known limited lifetime. Thus, when its lifetime expires, it can be swept by the lifetime manager.

GRIMOIRES is distributed as two dependent installations. A base installation covers all components in Figure 3, except the WSRF-compliant interfaces and the WS-Resource manager. Therefore, the base installation is a UDDI registry with metadata annotation capability that can be deployed into a WS container with or without WSRF support, for instance, Globus Toolkit 4 (GT4) [14] or an OMII [15] container. A WSRF installation extends the base installation with WSRF-compliant interfaces. Currently, the WSRF installation is targeted to GT4. Separating GRIMOIRES into two installations gives us flexibility of deployment in various WS container environments.

V. Implementation

To implement our vision of expressing UDDI service descriptions and their metadata annotations as WS-Resources, we faced several technical challenges, which we discuss in this section.

A. Cached Resources vs Registry State

The WS-Resource manager is in charge of exposing registry entities as WS-Resources. In order to maintain a view of WS-Resources consistent with registry entities, it has to *(i)* generate the `Entry` resources equivalent to registry entities; *(ii)* reflect into the `Entry` resources the changes of registry entities that are made through the non-WSRF-compliant interfaces; *(iii)* reflect into the registry the changes of the resources that are made through the `Entry` interface so that they can be seen by the non-WSRF-compliant interfaces as well; and *(iv)* save in the registry the resources that are created through the `Factory` interface.

In GRIMOIRES, resources act as yet another view of the registry data that is originally kept persistent as RDF triples in GRIMOIRES' triple store. Two options are available for the implementation of resources. Lasting resources could be kept available and consistent with corresponding registry entities, or transient resources could be generated on demand.

Implementing *lasting* resources introduces duplication of information between registry entities and resources, which increases the implementation complexity because we have to resolve the inconsistency between resources and registry entities. That is, on GRIMOIRES starting up, resources need to be initialized from the published registry entities in the triple store. Also, all changes made to the registry through the non-WSRF-compliant interfaces need to be captured, and corresponding resources then can be updated based on the captured events. On the other hand, the lasting resources in fact act as a cache of registry entities to the WSRF-compliant interfaces. Whenever requested, the resource is already available instead of being constructed from the triple store on the fly.

On the other hand, implementing *transient* resources does not cause duplication of information and does simplify the implementation, although the fact that resources have to be constructed on demand introduces some delay.

We have implemented both the lasting resource and the transient resource approaches and are experimenting with them. The choice between these two approaches appears to be essentially a tradeoff between time and space.

B. XPath Processing vs Translation

The `Query` interface provides for discovery over the whole registry, through the expressive XPath query language, but it presents some challenge from a performance viewpoint.

A possible optimization could be translating arbitrary XPath expressions to the query language used by the registry persistent storage. In our case, this means converting XPath queries into RDF queries. The corresponding RDF query expressions could then be directly evaluated over the triple store. Such a solution would avoid constructing resources at all, and it would leverage the RDF query engine that has been optimized to some extent.

However, a RDF query language such as RDQL has different expressiveness from XPath, which might make the translation difficult. This is a topic of investigation in itself and is beyond the scope of this paper. Instead, we decided on a two-pronged strategy. First, we opted for XPath processing and leveraged existing XPath solutions to address the problem. Second, we translated XPath queries that correspond to `UDDI` or `Metadata` canned queries and mapped them directly to the system programming layer.

C. XPath Processing

In order to prepare the `Query` resource properties document for XPath queries, `Entry` resources are generated from the RDF triples if transient resources are implemented; the `Query` resource is then constructed from `Entry` resources. By default, GT4 provides a query operation provider for evaluating XPath expression on a resource properties document, but it currently does not index the XML elements of the resource properties document to be evaluated, although such indexing could make the query more efficient.

To improve the performance of the `Query` interface, we have implemented our own query operation provider that overrides the default GT4 provider. GRIMOIRES query operation provider incorporates the following optimizations:

(i) The DOM representing the `Query` resource properties document, over which the XPath expression is evaluated, is cached in order to improve the performance of future invocation. The cached DOM is wrapped in a soft reference object so that it can be reclaimed by the garbage collector if the available heap space becomes scarce. The cached DOM is invalidated by updating the triple store.

(ii) A faster XPath query engine, Jaxen, is used instead of Xalan, the default in GT4. Note we do not do indexing in Jaxen either.

(iii) Some frequently used XPath expressions, for instance, discovering a service by its name, can be recognized and translated to corresponding `UDDI` or `Metadata` inquiry requests, which then are sent to the system programming layer directly. If the submitted XPath queries cannot be recognized, they are evaluated by using the XPath query engine over the DOM.

VI. Performance Evaluation

In this section, we evaluate the GRIMOIRES WSRF installation on GT4. GRIMOIRES' fundamental operations are service publication, metadata annotation, and service discovery (including service discovery by metadata). We wish to know the overhead of individual publication and discovery operations with respect to the data size of the registry. In particular, we wish to determine to what extent the publication and discovery overheads are affected when an increasing amount of data is registered in GRIMOIRES.

In our evaluation, we repeat the following procedure: (1) publish 100 services, each with its unique service description, and (2) among all the services currently registered in GRIMOIRES, randomly discover 100 services by name. We compare the overhead of the WSRF-compliant interfaces with that of the non-WSRF-compliant interfaces.

Figure 4 shows the service publication overhead. As seen from the figure, the overhead of the WSRF-compliant interface (i.e., by invoking the `Factory` interface) is 8.4% lower than that of the `UDDI` interface when there are 3,000 services in the registry. To publish a service through the `UDDI` interface, in our experiment, we first publish the business as the service provider, then publish the service, whereas through the `Factory` interface both business information and service information are published in one message. Two uncertainties cause the jumps along the curves in Figure 4: garbage collection and synchronizing data from the memory to the backup file. The latter is due to the file-backed in-memory triple store we have used in GRIMOIRES. All service descriptions reside in memory for better performance; after a variable delay they are saved to the backup file to ensure persistence.

Figure 5 shows the service discovery overhead. In order to discover a service by its name, a common question in service discovery, a WSRF-compliant client submits an XPath query through the `QueryResourceProperties` operation defined in the WS-ResourceProperties specification. As seen from the figure, the service discovery performance through the Xalan XPath query engine lags behind that through the `UDDI` interface. We therefore have made various efforts to improve the performance, namely, by using Jaxen XPath query engine instead of Xalan, caching the DOM representing the `Query` resource properties document, and translating the XPath query to the `UDDI` query. When there are 3,000 services in the registry, using Jaxen instead of Xalan improves the performance by 6.9%; caching DOM improves the performance by 91.7%; and translating the XPath query to the `UDDI` query makes the query performance very similar to that of the `UDDI` interface. We note that indexing currently is not used in the XPath query; we are investigating ways to improve the XPath evaluation performance by indexing frequently used DOM elements.

Figures 6 and 7 show the overhead of attaching or updating metadata and that of discovering service by metadata, respectively. The methodology is the same as in the previous experiment except that an item of metadata is attached to each service description during the publication phase and services are discovered by their metadata during the discovery phase.

To attach metadata to a service description, the corresponding `Entry` resource needs to be identified and its `AnnotatedRegistryEntity` resource property element needs to be retrieved. A new piece of metadata then can be added into this `AnnotatedRegistryEntity` element. Thus, attaching metadata requires two WSRF operations: `GetResourceProperty` and `SetResourceProperties`. Updating metadata is similar to attaching metadata except that an existing piece of metadata in the `AnnotatedRegistryEntity` element is modified. In comparison, attaching or updating metadata through the `metadata` interface requires only one Web service invocation.

In Figure 7, we again observe that the service discovery performance through the Xalan XPath query engine lags behind that through the `Metadata` interface. When there are 3000 services in the registry, however, using Jaxen XPath query engine instead of Xalan improves the performance by 2.4%; caching DOM improves the performance by 89.1%; and translating the XPath query to the `Metadata` query makes the query performance very similar to the `Metadata` interface.

VII. RELATED WORK

UDDI [1] has become the de facto standard for service publication and discovery for Web services. In a previous paper [4], we identified the incapability of UDDI in terms of metadata annotation, and we described GRIMOIRES' approach for supporting metadata annotation in the context of a UDDI registry. In this paper, we present an approach for exposing UDDI entities and their annotated metadata as WS-Resources. With such an approach, GRIMOIRES augments the service registry's interoperability in accessing service description and serving service lifecycle events. UDDI does not specify a registry entity lifetime management mechanism, which is now addressed in GRIMOIRES using WSRF. Although UDDI defines its own notification mechanism, GRIMOIRES' notification mechanism, which is built on the well-accepted WS-Notification, presents a better interoperability.

A UDDI technical note [9] describes a best practice to attach metadata (so called properties in the technical note) to UDDI entities, i.e., using the KeyedReference data structure that contains a key/value pair to express a piece of metadata. Several research papers ([16], [17]) investigate how to express semantics in UDDI based on such a mechanism. To assist service discovery by its technical interface, Sivashanmugam et. al. [17] suggest attaching a KeyedReferenceGroup data structure to a UDDI service for each service operation. Three KeyedReferences can be found inside the KeyedReference-Group, expressing the semantics of the operation, the operation input, and operation output, respectively. Compared with GRIMOIRES' metadata annotation approach, this mechanism limits metadata to key/value pairs, thus lacking the capability to express structured metadata. Also, it does not address the problem of allowing third party annotation. Furthermore, Sivashanmugam et. al.'s approach requires publishers to re-describe services' technical interface information explicitly in the UDDI data model. Comparatively, GRIMOIRES uses WSDL to describe service interface, and provides a mechanism to link a UDDI service to its corresponding WSDL description, which is a more natural way to describe service interface in Web Services development environment.

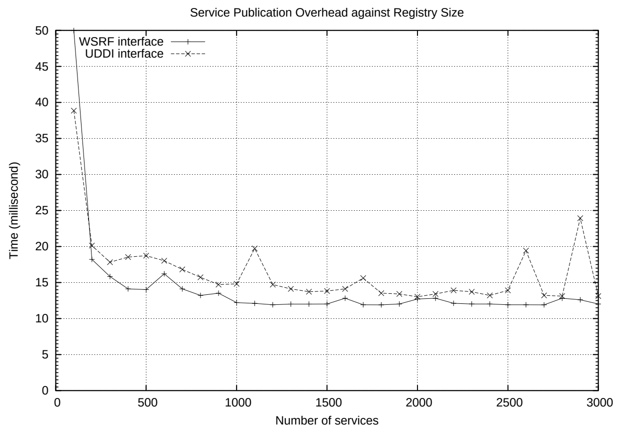

Fig. 4. Service publication

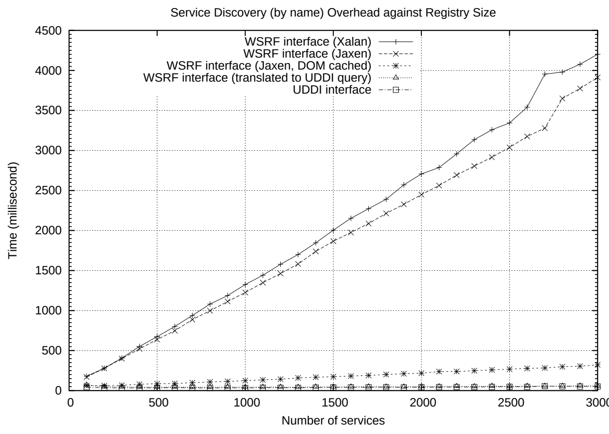

Fig. 5. Service discovery (by name)

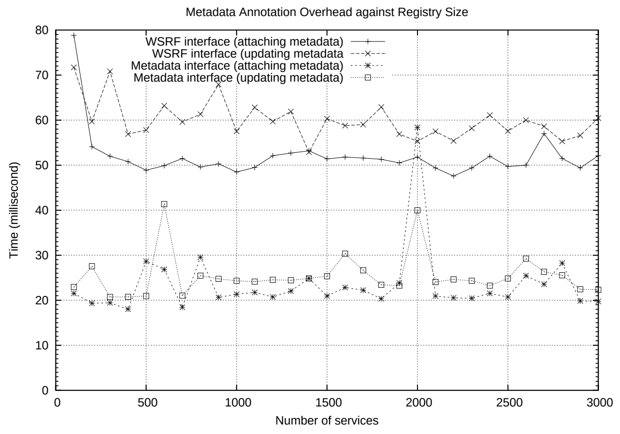

Fig. 6. Metadata annotation

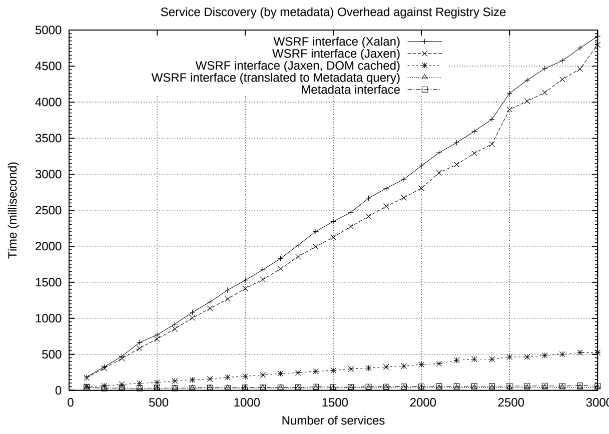

Fig. 7. Service discovery (by metadata)

The ebXML registry [2] is designed to be a generic information management system. It has defined both a registry information model and registry services and interfaces. The ebXML registry has provided a Web service profile that specifies the conventions to make the ebXML registry act as a Web service registry.

The Globus Toolkit (`www.globus.org`) provides the Monitoring and Discovery System (MDS), consisting of a suite of Web services, to monitor and discover resources and services on Grids. MDS focuses on the mechanism to disseminate and gather information on Grids rather than the information model to describe services or resources. Each information source publishes information in XML according to some schema. The GLUE schema [18] is used for compute information. But the authors of the information sources or the Grid resources can define their own schema, so that arbitrary XML data can be published to describe service profiles and states. The XPath language is used for inquiry. We consider MDS as a complement to GRIMOIRES in that MDS can be used as an automatic service information collector for the GRIMOIRES registry.

WS-ServiceGroup is a part of the WSRF specifications, which defines the conventions by which Web services and WS-Resources can be grouped together. The conventions include a data model for the service group, such as some common constraint rules among members, and interfaces for group membership management. GRIMOIRES uses a data model much richer than that defined by WS-ServiceGroup. GRIMOIRES supports many registry entities that cannot be modeled as services identified by an endpoint reference or a URL. For instance, in UDDI's taxonomy, neither BusinessEntity nor BusinessService needs to have a URL. Only the concrete binding of a service (i.e., BindingTemplate) must have a URL.

VIII. Conclusion and Future Work

In this paper, we present a novel service registry architecture, GRIMOIRES, that is UDDI-compliant and WSRF-compliant and strongly supports metadata annotation. Users can use either standard UDDI operations or standard WSRF operations to publish and discover service descriptions. The WSRF-compliant interface of GRIMOIRES further gives users the capability to track the service description changes by notification and to manage the lifetime of service descriptions.

Our prototyping and performance benchmarking endorse the feasibility and benefits of such an architecture. There is room for further optimizing the query performance through the WSRF-compliant interface. Indeed, by capturing and translating UDDI canned queries such as discovering a service by its name, the query performance through the WSRF-compliant interface is comparable to that through the UDDI-compliant interface, while the performance of other queries may suffer from the overhead of a flexible and expressive query language. We are investigating using indexing to further improve the query performance.

The query language XPath used in the WSRF-compliant interface does not support semantic reasoning, since it performs exact syntactic match over the XML document representing WS-resources. We are further investigating how such semantic reasoning could be supported.

Recently, a convergence of Web service standards for resources, events, and management [19] has been proposed by IBM, Microsoft, HP, and Intel. The new proposed standards will reuse core concepts of existing ones, such as WSRF. Our approach, which is built on WSRF, would migrate easily to the future standards.

ACKNOWLEDGMENT

This research is funded in part by the GRIMOIRES (EPSRC Grant GR/S90843/01) and SOCA (EPSRC Grant EP/C528131/1(P)) projects. This work was supported in part by U.S. Department of Energy, under Contract W-31-109-Eng-38.

REFERENCES

[1] *The UDDI Specification*, OASIS Std., Rev. Version 2, 2002. [Online]. Available: http://www.uddi.org

[2] *The ebXML Registry Standards*, OASIS Std., Rev. 3.0, 2005.

[3] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid information services for distributed resource sharing," in *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, 2001.

[4] S. Miles, J. Papay, T. R. Payne, K. Dceker, and L. Moreau, "Towards a protocol for the attachment of semantic descriptions to grid services," *Scientific Programming*, vol. 12, pp. 201–211, 2005.

[5] I. Foster, K. Czakowski, D. F. Ferguson, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke, "Modeling and managing state in distributed systems: The role of OGSI and WSRF," vol. 93, pp. 604–612, 2005.

[6] *Web Services Base Notification*, OASIS Working Draft, Rev. 1.2, 2004.

[7] I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czakjowski, D. Ferguson, F. Leymann, M. Nally, I. Sedukhin, D. Snelling, T. Storey, W. Vambenepe, and S. Weerawarana, "Modeling stateful resources with web services," 2004.

[8] *Modeling and Managing State in Web Services Resource Properties*, OASIS Std., Rev. 1.2, 2006.

[9] L. Clément and J. Garbis, *Strawman Proposal - Representing Property Information*, OASIS Technical note proposal, 2005.

[10] F. Curbera, D. Ennebuske, and D. Rogers, *Using WSDL in a UDDI Registry*, OASIS Technical note, 2002.

[11] *RDF Primer*, WWW Consortium Recommendation, 2004. [Online]. Available: http://www.w3.org/TR/rdf-primer

[12] V. Tan, W. Fang, S. C. Wong, S. Miles, and L. Moreau, "A security architecture for a semantic grid registry," in *Proceedings of the UK OST e-Science Fourth All Hands Meeting 2005 (AHM'05)*, Nottingham, UK, 2005.

[13] *Web Services Resource Lifetime*, OASIS Std., Rev. 1.2, 2006.

[14] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," in *Proceedings of IFIP International Conference on Network and Parallel Computing*, 2005.

[15] Open middleware infrastructure institute. [Online]. Available: http://www.omii.ac.uk/

[16] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Importing the semantic web in uddi," in *Proceedings of Web Services, E-Business and Semantic Web Workshop*, Toronto, Canada, 2002, pp. 225–236.

[17] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller, "Adding semantics to web services standards," in *Proceedings of The 2003 International Conference on Web Services (ICWS'03)*, Las Vegas, NV, 2003, pp. 395–401.

[18] The GLUE schema. [Online]. Available: http://www.cnaf.infn.it/˜sergio/datatag/glue

[19] K. Cline, J. Cohen, D. Davis, D. F. Ferguson, H. Kreger, R. McCollum, B. Murray, I. Robinson, J. Schlimmer, H. Shewchuk, V. Tewari, and W. Vambenepe, "Toward converging web service standards for resources, events, and management," A Joint White Paper from Hewlett Packard Corporation, IBM Corporation, Intel Corporation and Microsoft Corporation, 2006.