Semantic and Personalised Service Discovery

Phillip Lord[1], Chris Wroe[1], Robert Stevens[1],Carole Goble[1], Simon Miles[2],
Luc Moreau[2], Keith Decker[2], Terry Payne[2] and Juri Papay[2]
[1]Department of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL
UK
and
[2]School of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ
UK
p.lord@russet.org.uk,
[chris.wroe,robert.stevens,carole]@cs.man.ac.uk,
[sm,l.moreau,ksd,trp,jp]@ecs.soton.ac.uk
http://www.mygrid.org.uk

Abstract

One of the most pervasive classes of services needed to support e-Science applications are those responsible for the discovery of resources. We have developed a solution to the problem of service discovery in a Semantic Web/Grid setting. We do this in the context of bioinformatics, which is the use of computational and mathematical techniques to store, manage, and analyse the data from molecular biology in order to answer questions about biological phenomena. Our specific application is *my*Grid (http://www.mygrid.org.uk) that is developing open source, service-based middleware upon which bioinformatics applications can be built. *my*Grid is specifically targeted at developing open source high-level service Grid middleware for bioinformatics.

1 Introduction

Service discovery, the process of locating services, devices and resources, is an essential requirement for any distributed, open, dynamic environment. Although traditional service discovery methods may be effective when *a priori* knowledge of the services or agreements about implicitly shared ontologies can be assumed, they fail to scale to large, dynamic, open, environments, where a high degree of autonomy is required. Semantic web service discovery overcomes this limitation by providing an ontological framework by which services may be described and processed. Whilst this is equally applicable to Grid and e-Science domains, these domains impose additional requirements on the service discovery process, beyond simply locating a service based on a description of its functionality. This paper examines the issues, and proposes a hybrid solution to the task of semantic web service discovery within the context of a Bioinformatics Grid domain. This domain uses computational and mathematical techniques to store, manage, and analyse data from molecular biology in order to answer questions about biological phenomena. Our specific application is *my*Grid (http://www.mygrid.org.uk).

Molecular biology is about collecting, comparing and analysing information from experimental data sets. Traditionally, these (typically small) data sets are manually obtained from specific "wet" bench experiments designed to test a specific hypothesis. *In silico* experimentation has allowed molecular biologists to obtain relatively large datasets, by conducting experiments purely through computer based analysis of existing experimental data and associated knowledge to test a hypothesis, derive a summary, search for patterns or to demonstrate a known fact. Thus, experiments can be performed on a complete genome rather than an individual gene; to model the behaviour of a cell's complement of genes, rather than one gene; and to compare between species rather than within

one particular species. This form of e-Science involves marshalling disparate, autonomous, and heterogeneous resources to act in concert to achieve a particular analytical goal.

Bioinformatics resources, such as experimental data, services, descriptions of experimental methodology, are knowledge-rich and require a great deal of semantic description for pragmatic use, even within semi-automated processes. They should support third-party annotations, which may have limited visibility or scope. For example, a scientist may need to record additional comments with these resources whilst performing an experiment, such as the applicability of a service for a given context, and share these comments only with immediate colleagues. Several such additions may be generated by different third-parties.

^{my}Grid is specifically targeted at developing an open source, high-level service, Grid middleware for this kind of biology. ^{my}Grid middleware is a framework using an open, service-based architecture, prototyped on Web Services with a migration path to the Open Grid Services Architecture (OGSA) [3]. The key aim is to support the construction, management and sharing of data-intensive *in silico* experiments in biology. In order to achieve this the ^{my}Grid middleware explicitly captures the experimental method as a *workflow*. The use of data/computational services and the derivation of experimental data is tied to the corresponding workflows by explicit *provenance* information. Figure 1 shows the lifecycle of *in silico* experiments, along with the core activities of ^{my}Grid. Resource discovery pervades the life cycle. Before developing an experimental method in the form of workflow the user should be supported in re-using and adapting previous work in the community rather than having to start from scratch.

All these activities can involve *discovery* – for example, "who has performed an experiment *x*, when, where and why?", a question involving details of provenance, location, experimental method, etc. Data and computational services need to be discovered so that they perform individual tasks in the workflow. In fact there is nothing to stop these tasks being performed by more detailed workflows, rather than a single service.

1.1 Service Semantics

Semantic description of implicit community knowledge offers a mechanism to cope with the heterogeneity of resources by providing a rich descriptive framework and common vocabulary to integrate and search over apparently disparate data, services and workflows. Several discovery services have been deployed

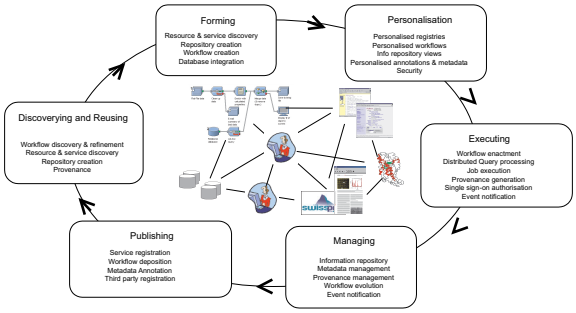

Figure 1: The cycle of ^{my}Grid *in silico* experiments.

that utilise description logic reasoning to match a request against different advertised service profiles systems [6, 2]. This provides flexibility within the matching algorithm, allowing the search to be broadened to services that consume more general inputs or produce more specific outputs. Within ^{my}Grid we have also based semantic service descriptions on the DAML-S profile schema with specific extensions for bioinformatics [7]. However, we have decided not to force service publishers and third parties to describe business details, workflow or binding using the schema provided by the DAML-S upper level ontology, Instead, industry standards and associated tools can be used to author and discover such information. In ^{my}Grid these include the UDDI model for specifying business details, Web Services Flow Language (WSFL) for workflow, and WSDL for binding information. This lowers the entry cost for publishing or annotating a service. The DAML-S based approach is only used for semantic discovery where domain ontologies (such as bioinformatics ontologies) and associated reasoning are essential.

In Section 2 we analyse the requirements of the *in-silico* bioinformatics domain and present our architecture to meet those requirements in Section 3. Exactly how the components of the architecture interact to solve the service discovery problems is discussed in Section 4. We conclude in Section 5.

2 Requirements for Publishing and Discovering Services

Service discovery is a process in which a user or other agent gives a query to the system and is presented with a list of available services that match that query. The query will state what the user wishes to achieve or what data they wish to process or service he or she wishes to discover more about.

The nature of the bioinformatics community (as described above) presents ^{my}Grid with several interesting challenges: Global distribution and high frag-

mentation of community (except for a few centralised repositories); autonomy of community groups (over 500 resources are available at the time of writing); autonomy of applications, services and formats that lead to massive heterogeneity.

The different community groups produce a range of diverse data types such as proteomes, gene expressions, protein structures, and pathways. The data covers different scales and different experimental procedures that may be challenging to inter-relate. The different databases and tools have different formats, access interfaces, schemas, and coverage, and are hosted on cheap commodity technology rather than in a few centralised and unified super-repositories. They commonly have different, often home-grown, versioning, authorisation, provenance, and capability policies.

Within bioinformatics we cannot assume that we have control over the format data presented by the services. Many service providers will therefore be unwilling to represent their data according to a "standard" representation, preferring to use either their own formats, or one of the existing, hard won, bioinformatics standards. Additionally the complexity of biological data means that we may wish to describe a piece of data in several different ways, e.g. Two services might both return a DNA sequence, but one might be a complete genome, the other might return only single genes, information which is not easy to explicitly encode in a WSDL interface. It is for this reason that, within ^{my}Grid, we have investigated semantic web technologies.

We start, in the section below, by presenting examples of the types of query that may be presented by users in our domain.

2.1 Sample Queries

In order to design the discovery architecture for ^{my}Grid we have collected an example set of questions and categorised them depending on the nature of the information that must be searched.

The first category consists of queries which involve searching on the properties of a service or workflow resource as described by the publisher in terms of concrete instance data, such as finding a resource based on its ownership, location, or accessibility. Examples include:

- What resources does a specific organisation provide?
- Who authored this resource?

This requires the author of services to describe these properties using a consistent schema. For example, businesses and services can be described in UDDI using a standard data model. Such a description must be available to the discovery service at the time of registration of the service or publication of a workflow. A discovery service must then be able to process queries over these descriptions. In this case the type of descriptive information is common to any domain to which the service is targeted. For example, organisation, authorship, location, address, etc. are features of any domain within e-Science or business.

The second category consists of queries which involve searching on concrete instance based properties provided by third parties (users, organizational administrators, domain experts, independent validating institutions, etc.) either as opinion, observable behaviour or previous usage.

- What services offering x currently give the best quality of service?
- Which service would the local bioinformatics expert suggest we use?

Figure 2 shows an example of third party description of a resource conforming again to the DAML-S profile schema.

```
<profile:qualityRating>
 <profile:QualityRating rdf:ID="NCBI-BLASTn-Rating">
  <profile:ratingName>Recommendation</profile:ratingName>
  <profile:rating rdf:resource="http://www.mygrid.org.uk/quality_concepts.daml#recommended"/>
 </profile:QualityRating>
</profile:qualityRating>
```

Figure 2: RDF based description of author and publishing organisation adhering to the DAML-S service profile

The need for third party description immediately introduces the requirement for control of who is permitted to describe a resource and proper attribution of a description to an author. It would be desirable to allow local (organizational and personal) annotation of resources registered in global registries. Another consequence of third party annotation are views based upon those third party annotations. Individuals, groups, communities and institutions may differ in their opinions of a service.

The final category consists of queries which involve searching over properties expressed using concepts from a domain specific ontology.

1. Finding a service that will fulfil some task e.g. aligning of biological sequences.
    - What services perform a specific kind of task, for example, what services can I use to perform a biological sequence similarity search?

2. Finding a service that will accept or produce some kind of data.

    - What services produce this kind of data, for example, from where can I find sequence data for a protein?
    - What services consume this kind of data, for example, if I have protein sequence data, what can I do with it?

An example of a commonly used domain service in bioinformatics is BLAST– "the Basic Local Alignment Search Tool" [1]. It is an application that encompasses a number of services used to compare a newly discovered DNA or protein sequence with the large public databases of known sequences. It can therefore accept as input a variety of sequence data whether protein or DNA, perform a search over a variety of databases and produce a variety of result formats. Figure 3 shows a conceptual description of the BLAST service BLASTn in DAML+OIL. At its core it accepts nucleotide sequence data and compares this against nucleotide databases. It is a common situation for the user to actually have a more specific type of data such as an Expressed Sequence Tag (EST), which is a fragment of DNA known to be derived from a gene. To successfully answer the query "what service will accept an expressed sequence tag?", it is necessary for the discovery service to have information about the *domain* describing the semantic relationships between the bioinformatics datatypes. In *my*Grid this domain information is stored as a suite of domain ontologies [7]. It should also be clear that users may wish to search for resources, other than services, with these same semantic relationships. So as well querying for "all services taking DNA sequences", we may wish to ask for "all local files containing a DNA sequence".

```
class-def defined BLAST-n_service
   subclass-of service
   has_Class performs_task (aligning has_Class has_feature local has_Class has_feature pairwise)
   has_Class produces_result (report has_Class is_report_of sequence_alignment)
   has_Class uses_resource (database has_Class contains
                (data has_Class encodes
                        (sequence has_Class is_sequence_of nucleic_acid_molecule)))
   has_Class requires_input (data has_Class encodes
                (sequence has_Class is_sequence_of nucleic_acid_molecule))
   has_Class is_function_of (BLAST_application)
```

Figure 3: DAML+OIL description of the functionality of BLASTn

This categorisation of queries will not be obvious to the user and indeed a single user query may incorporate all the aspects we have described simultaneously. For example 'Which services recommended by my organisation can I use to process my expressed sequence tag?' Therefore, although it may be essential for the architecture to separate out ontology based queries from queries of third party descriptions from queries on original published information, it is also essential to shield the user from such a distinction.

2.2 Requirements Summary

We would argue that the following requirements, over and above the generic requirements of web services, are necessary to support service discovery in an e-Science context:

1. Descriptions must be attached to different resources (services and workflows) published in different components (service registries, local file stores, or databases);
2. Publication of descriptions must be supported both for the author of the service and third parties;
3. Different classes of user will wish to examine different aspects of the available metadata, both from the service publisher;
4. There is a need for control over who make add and alter third party annotations;
5. We must support two types of discovery: the first using cross-domain knowledge; the second requiring access to common domain ontologies;
6. A single, unified interface for all these kinds of discovery should be made available to the user.

3 Architecture

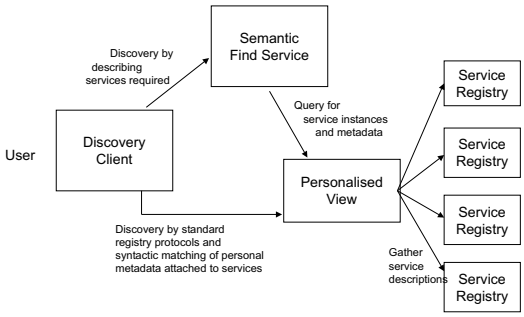

Figure 4: Architecture of discovery services in *my*Grid

In this section, we discuss the *my*Grid architecture used to support the types of service discovery discussed in the previous section; Figure 4 shows the relevant components. We assume that there exist a

multitude of service registries on the Grid which can be used to publish details on how to access services, possibly with additional information to aid discovery.

In order to allow service discovery using third party metadata, we need a place to store that metadata. Metadata may be personal and private to an individual or organisation and so should not be published in public registries, even if that was technically possible. Third-party metadata intended to inform service discovery is one way in which to filter the services returned to a user on providing a query. A *personalised view* is a service that provides a place to add third-party metadata and thereby filter the service details returned by a query. Information from registries is collected into personalised views that provide a subset of service advertisements that can be annotated with metadata by an individual or organisation and then used for discovery.

Semantic find services use the information (and in particular the metadata) stored in views to extract relevant semantic descriptions of services allowing semantic discovery using domain knowledge. A discovery client can be used by a user to hide the distinctions between the syntactic matching performed by the view and the semantic reasoning done by a find service.

3.1 Service Registries

Services can currently be advertised using a variety of standards, e.g. LDAP, Jini. Within ^{my}Grid, we have mostly been concerned with Web Services, for which the primary publishing "standard" is UDDI. UDDI repositories can be deployed on the Internet for general use, or privately within an organisation as repositories of that organisations' own services. A UDDI repository contains a set of adverts for services, each of which is usually registered by the provider of the service. Service descriptions follow a strict data model including information such as the organisation owning the service; details on how to contact the service; references to technical information regarding the interface of the service; simple classification of the service within some standard taxonomy etc.

However, this simple model is inadequate for meeting the demands of ^{my}Grid as set out in Section 2, as there is no semantic reasoning, no third-party metadata and only simple classification.

Registries are necessary for allowing existing service discovery to take place. Using these registries we can solve the problem of users being able to locate services that might match their needs by browsing registries for organisations providing such services. Standard registries provide the functionality for cross domain queries discussed in Section 2.

3.2 Views

A *view* is a service that allows discovery of services over a set of service descriptions stored in directories on the grid. The discovery process can be personalised by attaching third-party metadata to service descriptions. An (experienced) user can set up a view that pulls entries from a set of sources (registries). For each source, the user specifies a query to provide the initial data extracted from that source. Third parties can manually edit the view by editing the metadata attached to entries or deleting entries.

A view may be created and owned either by a single person or a organisation/group. For example, a biology lab could have a view that contains metadata useful to members of that lab and has one (or more) designated curator(s) authorised to change the view's entries and sources. A PhD student who joins a lab will be given access to the lab view of usable services. In their training period, the student will only be given read access to these views. At a later stage, the PhD student can have a view created for them by the view curator, with the lab view as its sole source, to which they can add metadata but make no other modifications. Later on, the view authorisation policy can be changed to allow them more control, such as modifying metadata and adding sources. Eventually, the PhD student can graduate to become the curator of the lab view.

The internal architectural details of views and how they can be used to store semantic information is described in [5].

One of the sample queries in the "third party" category in Section 2 is:

- Which service would the local bioinformatics expert, suggest we use?

A simple example of solving this problem is to have a view local to the organisation, and a piece of metadata attached to some service descriptions in the view. The metadata could have the name 'isRecommended' and either 'true' or 'false' as a value. The local bioinformatics expert can attach this metadata to the services described in the view that they favour. Others in the organisation can then present a query that syntactically matches only those services with metadata of name 'isRecommended' and value 'true'. This provides a locally administered filtering of service discovery and also allows annotation of service descriptions.

3.3 Semantic Find Service

The *semantic* find service provides discovery over domain specific descriptions by reference to domain ontologies. The find service makes use of several ad-

ditional components as shown in Figure 5. The description database holds semantic descriptions gathered from resources published in registries and views. The ontology server provides access to the domain ontologies and manages interaction with the description logic reasoner FaCT [4]. The find service itself is responsible for:

- gathering semantic descriptions from the view and maintaining a reference back to the entry in the view, so that details for communicating with the services can later be retrieved;
- using the ontology service and associated reasoner to index items in the descriptions database to ensure efficient retrieval of entries at time of discovery;
- using the pre-built index or if necessary the ontology service and associated reasoner to process a discovery query

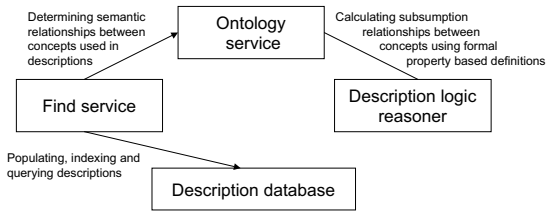

Figure 5: Internal architecture of the semantic find service

If we take the example of the BLASTn service presented in the requirements section we can demonstrate how the semantic find service can support a semantic query over such a resource description. The user presents a discovery query in terms of a DAML+OIL description of the kind of service they require. In the example case it could be a service which accepts Expressed Sequence Tags. The find service uses the ontology server to determine which services accept Expressed Sequence Tags or a more *general* semantic data type. The find service allows users to resolve queries of the "domain specific" category in Section 2.

The separation of the semantic service discover from registration stems from several key requirements. Firstly it enables the UDDI registration process, and semantic service advertisement to be providing by different people, i.e third party metadata. Secondly it allows substantial reuse of the semantic find service for discovery of entities other than services, such as workflows, or static data.

Finally it enables other service discovery techniques to be added. So, for example, imagine we wished to add a service which allowed discovery of bioinformatics services based upon some complex logic operating over the recommendations by third party bioinformaticians and the user's trust in those recommendations. So, the scalable ^{my}Grid architecture allows the addition of discovery mechanisms over a wide variety of metadata, as well as semantic advertisements.

3.4 The Discovery client

The discovery client guides the user in constructing a query that will adhere to the information model of service descriptions in ^{my}Grid and the ontology used to describe the domain specific semantic description of a services functionality. The user is presented with a form based interface which transparently integrates semantic and non semantic items of a query. The discovery client then separates the user request into the parts relevant for submission to either the semantic find service or view. It displays the intersection of the two queries to the user.

The discovery client removes the need for a user to have pre-existing knowledge of the data model or domain ontologies used to describe services. It also shields the user from having to know where to send specific components of their query and pooling the results. By providing this abstraction, queries of all categories in Section 2 are resolvable by the user.

3.5 Architecture and Requirements Summary

In summary the architecture meets the requirements given in Section 2.2, in the following ways.

1. Decoupling of service registration, and description, enables discovery over many entities (Requirement 1).
2. Providing a view over registries enables third party metadata, (Requirement 2), for discovery over subsets of total metadata (Requirement 3), and for controlling who can alter such metadata (Requirement 4).
3. The discovery client enables discovery of several kinds (Requirement 5), but with a single unified interface (Requirement 6).

4 Publishing and Discovery

Using the architecture presented in the preceding section, service providers can publish descriptions of their services and others can annotate those descriptions. This information is then accessible by users and can be searched over by presenting queries to the find services, views or registries.

Users of our architecture can attach, retrieve and reason over any published metadata such as services' ownership, location, recommendations, function, inputs or outputs. Public metadata will be stored in the registries, while private metadata is stored in the views owned by an organisation or individual biologist.

4.1 Publishing Service Descriptions

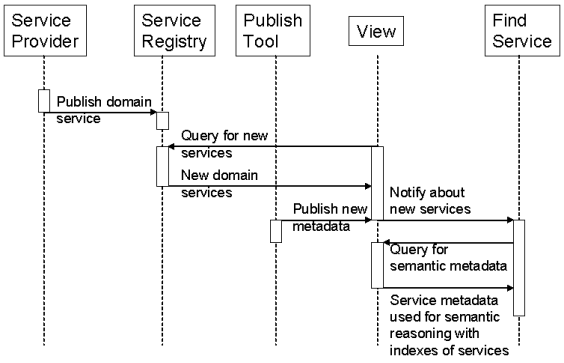

Figure 6: Sequence diagram of publishing service

UDDI and other registries have standard interfaces for publishing service descriptions following their own data models. Views allow users to attach metadata to any part of the service descriptions gathered from registry sources. Semantic data following the vocabulary and schema of a given ontology is gathered from views, and potentially other sources, and optimised for reasoning over.

Figure 6 shows the process that takes place when a service is published in the *my*Grid architecture. A service provider publishes their service in a registry on the Grid. The data is later pulled into views set up to monitor the registry, and a notification of the new service is sent to find services that have registered an interest. A find service can then query a view for the metadata attached to the service which provides information for semantic reasoning. The metadata is associated with service keys (indexes) that can later be used to retrieve communication information for clients to access the services.

4.2 Service Discovery

In Figure 7, we show the process of service discovery supported by our architecture. The user will provide a query to the system using the discovery client. This client divides up the query into the part requiring semantic reasoning handled by a find service, and the part using the data stored in a view. The find service has processed metadata containing semantic information extracted from the view into a form suitable for

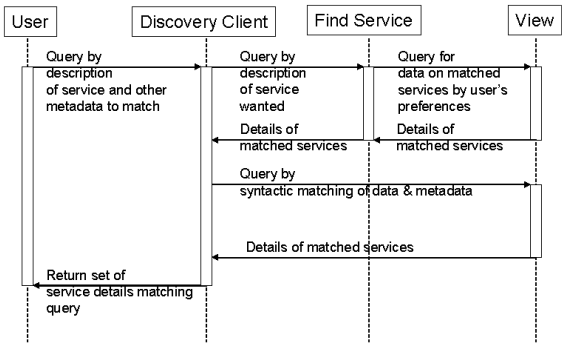

Figure 7: Sequence diagram of service discovery

reasoning over. The find service resolves the query results into a set of keys for extracting contact information (endpoints) of services from the view. The set of service instance information matching the query is returned to the discovery client and the user is provided with the intersection of these results and the ones returned by the direct query to the view.

The user may for example, wish to discover a service that accepts a gene sequence as input. A service description may not specify that it has an input exactly as a gene sequence, but may use a more specific concept for which semantic reasoning would be required to identify the data as suitable for providing as input. The metadata describing the service as taking a type of gene sequence as input would be contained in the view and extracted by the find service and analysed before discovery takes place. Other data and metadata stored in the views could be used directly to satisfy user preferences, such as recommendation of a service by a colleague or to limit the hosting organisation of the service. In the former case, the metadata would be personal to an organisation's view. In the case of the hosting organisation, this data would have been extracted from a registry on the Grid.

5 Conclusions

In this paper we set out our approach to solving some problems of service discovery in bioinformatics, by producing a flexible and scalable approach that: enables semantic descriptions of different types of entities, not just services; allows descriptions to be authored and stored in different places, not just a service registry; permits different abstractions of services, not just instances; and enables descriptions to be searched in different ways, not just by reasoning and classification.

By providing a flexible method of metadata storage in views, a variety of semantic descriptions can be attached to service advertisements as well as to the

input and output parameters of those services. This substantially extends the ability of existing registries as well as allowing annotation of personal metadata by the user. Find services provide a discovery mechanism over the metadata in views and descriptions of other entities, such as the data produced by an experiment, stored in other repositories. Find services, using ontologies for vocabularies and schemas, allow abstraction over services and other concepts, and so can provide a very rich querying and discovery mechanism.

Acknowledgements

This work is supported by the ^{my}Grid e-Science pilot project grant (EPSRC GR/R67743) and the GONG project grant (DARPA DAML subcontract PY-1149 from Stanford University).

References

[1] S.F. Altschul, W. Gish, M. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[2] Ion Constantinescu and Boi Faltings. Efficient matchmaking and directory services. Technical Report 200277, Ecole Polytechnique Federale De Lausanne, 2002.

[3] Ian Foster, Carl Kesselman, Jeffrey Nick, and Steven Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. *globus*, 2002.

[4] I. Horrocks. FaCT and iFaCT. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, pages 133–135, 1999.

[5] Simon Miles, Juri Papay, Vijay Dialani, Michael Luck, Keith Decker, Terry Payne, and Luc Moreau. Personalised grid service discovery. In Stephen A. Jarvis, editor, *Performance Engineering. 19th Annual UK Performance Engineering Workshop (UKPEW 2003)*, pages 131–140, University of Warwick, UK, July 2003.

[6] Massimo Paolucci, Takahiro Kawamura, Terry Payne, and Katia Sycara. Semantic matching of web services capabilities. In *The First International Semantic Web Conference (ISWC)*, 2002.

[7] C. Wroe, R. Stevens, C. Goble, A. Roberts, and M. Greenwood. A suite of DAML+OIL ontologies to describe bioinformatics web services and data. *International Journal of Cooperative Information Systems*, 12(2):197–224, 2003.