

# Feta

**DOI:**
10.1007/11431053_2

**Document Version**
Accepted author manuscript

# Feta: A light-weight architecture for user oriented semantic service discovery

Phillip Lord, Pinar Alper, Chris Wroe, and Carole Goble

School of Computer Science,
University of Manchester,
Oxford Road,
Manchester
M13 9PL
UK
p.lord@cs.man.ac.uk

**Abstract.** Semantic Web Services offer the possibility of highly flexible web service architectures, where new services can be quickly discovered, orchestrated and composed into workflows. Most existing work has, however, focused on complex service descriptions for automated composition. In this paper, we describe the requirements from the bioinformatics domain which demand technically simpler descriptions, involving the user community at all levels. We describe our data model and light-weight semantic discovery architecture. We explain how this fits in the larger architecture of the $^{my}$Grid project, which overall enables interoperability and composition across, disparate, autonomous, third-party services. Our contention is that such light-weight service discovery provides a good fit for user requirements of bioinformatics and possibly other domains.

## 1 Introduction

Web Services and Service Orientated Architectures offer the possibility of the composition and orchestration of distributed resources. The $^{my}$Grid project has been seeking to apply these technologies to the bioinformatics domain. To this end, it has contributed to the generation of a rich service layer, has built a workflow engine and workflow development environment [15].

However, there is a difficulty. Even with a good environment, producing complex workflows is difficult, time-consuming and expensive. One of the reasons for this is the shear number of services which are available for the biologist to use. Commonly, to promote reusability, each bioinformatics service provides a small unit of functionality. A useful task is achieved by combining these services into a workflow. Scientists using $^{my}$Grid currently have access to over 1000 bioinformatics services. In a research domain where many workflows are developed in-house, experimented with, then either thrown away or extensively modified, speed of development is a rate limiting step [16].

There has been a large amount of interest in "Semantic Web Services". In board outline, this augments standard Service Oriented Architecture with se-

mantic descriptions of the services. These descriptions help agents (whether human or machine) interact with the service during its life cycle including discovery, composition, execution and monitoring. Considerable work on technologies such as OWL-S [8] and, more recently, WSMO [1], have focused largely on descriptions to enable automated composition of services. The requirement for full automation, and transparency of composition from the user perspective, has resulted in the application of extremely rich service descriptions, using description logics, or F-logic, based formalisms. In keeping with these approaches, the $^{my}$Grid project has built a domain ontology for describing bioinformatics services, described services semantically using that ontology, and used description logic reasoning to support service discovery [19]. However, we found this approach bought considerable complexity to the architecture, without addressing the requirements of our target users (biologists and bioinformaticians). This complexity arose from at least three areas:

1) **Service Descriptions** There are many aspects to a service which can be described at varying levels of detail, depending on the intended audience and use of the description[18]. The description needed to discover a service is different from that needed to configure a service, which may be different from that to invoke a service. Descriptions for unattended agents need to be more formal and explicit than if there is human user involvement. OWL-S recognizes some of this variety by providing a profile for discovery, a process model for orchestration, and a binding for invocation [8]. However, we have found it is not realistic for service providers to provide comprehensive service descriptions as suggested by OWL-S. Sections 2 and 3 examine how the requirements from the bioinformatics discipline allow us to simplify the problem – narrowing the use of description to discovery, and assuming a human is always responsible for final selection.
2) **Variety of Services** The services available for building workflows are heterogenous. Only a small proportion could be classed as plain web services (i.e. Web Services described using a Web Services Description Language (WSDL) document with no additional conventions). More information on this heterogeneity is given in section 4. We have found that providing detailed formal descriptions of how each type of service varies in its behaviour is unrealistic. Section 5 illustrates how we simplify the problem by describing only an abstraction of a service that captures its functionality (as recognized by the workflow builder) whilst omitting its more technical behaviour. Section 4 explains how other components (particularly the workflow enactor Freefluo) bridge the gap, and loosely bind the abstract description to concrete implementation.
3) **Reasoning** The use of formal semantics within service descriptions allows the use of reasoning. For example, discovery of a service can be supported by description logic (DL) reasoning that matches a requirement for a service against a set of available services. It does so by recasting it as a classification task and testing for those services that are *subsumed by* the requirements. Our experience in implementing such a DL based system is that sophisticated rea-

soning is computationally expensive, a cost that is not justified by the benefits it provides to users searching for services. The complexity of deploying reasoning services within the $^{my}$Grid middleware also proved costly. Section 6 describes our discovery component Feta, which allows users to search over simple semantic descriptions of services using simpler reasoning components.

This paper describes our solution for managing (in our experience) the prohibitive complexity of deploying a fully featured semantic web services architecture. By adopting semantic web technologies such as RDF and OWL yet drastically cutting back on the features used, we are able to deploy a manageable middleware system. As both the technology and the tools progress, we leave the door open for adding complexity back into the system, but only as required by users.

## 2 The Application of Service Orientated Architectures

Bioinformatics involves the application of computational tools to the problems of biology. It is largely reflective of its history, originating in a large number of small molecular biology laboratories. Each lab generally investigates a small area of biology, with only a few labs world-wide working on a particular area. The data and tools generated by these labs has been made available for the community by web publication. More recently, large quantities of data and many tools have been developed and released by relatively few genome centres.

The use of the web as a primary means of data publication has obvious difficulties: HTML enables presentation to humans, not formal data modelling; services are hard to find; interoperability is poor. The primary mechanism for overcoming these problems has been the expert biologist; cutting and pasting between web delivered forms has been the norm [14]. Automated tools for the service composition have largely been built over the top of these web delivered services, often using Perl, and screen-scraping techniques. While this works, it tends to be fragile to changes in the website.

It is against this background that the $^{my}$Grid project has operated. Along with other projects [7], $^{my}$Grid has developed more formal, programmatically accessible middleware, to enable transfer of information and composition of data and tool services into large workflows, addressing the real needs to the lab biologist [15]. This environment comprises of a number of different components including: 1) Soaplab–A toolkit for the presentation of legacy command line applications, which covers the majority of bioinformatics tools, as Web Services. 2) Taverna–A workflow construction environment which enables the composition of Soaplab and other services into, often large and complex, workflows. 3) FreeFluo–A workflow enactment environment which enact Taverna workflows, invoking services, gathering information and returning it to the user.

The $^{my}$Grid environment has simplified the task of accessing the data sets that are available; however, it has been a victim of its own success. At the current time, there are over a 1000 services which can be used by $^{my}$Grid. This leaves the end user with a substantial problem in terms of selecting appropriate services

for use. This is made worse as the user, in this case, is the biologist who may not be highly skilled or knowledgeable about these services [16].

$^{my}$Grid assumes the contributions of third parties. Initially most of the tools available for use came from within the project, deployed via Soaplab. Currently, the larger part of the services in use come from external, autonomous service providers. This means there is no unified $^{my}$Grid type system determining the structuring of the data passed between services. As a result, while $^{my}$Grid has reduced some of the previous fragility, it has not solve the difficulties of interoperability.

On the face of it, therefore, the problems of $^{my}$Grid seem to be closely aligned to those addressed by the Semantic Web and SW Services. We have a set of services which we wish to compose in ways unanticipated by the providers or those responsible for the middleware.

## 3 Semantic Web Services in Bioinformatics

While the problems of bioinformatics appear to be closely aligned to those addressed by current SWS efforts, a consideration of the particular nature of the domain suggests to us that this is not the case, due to a set of constraints placed upon us by the domain. In this section, we consider these constraints.

**User Transparency:** Existing frameworks have focused on the requirement for transparent composition of web services in order to achieve the high level goals given by users (e.g. Make my travel arrangements for the next WWW conference). While this level of automation may be appropriate in B2C applications, it is less desired within bioinformatics, where the user base wish to be involved in service selection. There are two main reasons for this. Firstly, bioinformatics is a scientific endeavour and much depends on the correct selection of services, something that the users may later be forced to justify under peer review. Secondly, they suspect that they will make better decisions than a software agent. In this they are probably correct. Service selection requires significant areas of knowledge, including the technology of bioinformatics, the biological questions being examined and the level of trust the user has in different data sources. Any model of bioinformatics is likely to be wrong, at least in the first instance, particularly if the model is built by the middleware providers rather than those with the biological knowledge.

**Messaging Opacity:** One of the key difficulties with integration in bioinformatics has been the lack of formal and explicit structuring for its key datatypes. This is true for even relatively simple datatypes, such as DNA sequence which is a simple two-bit code; there are at least 20 different flat file formats for representing this data. The standards that do exist have often come about as a result of many years of collaborative work, so both service providers and consumers have a large investment in these (in)formalisms. Changing these data formats is not an attractive option. The practical upshot of this is that most web services provided for use within bioinformatics do not formally describe their messaging formats with a WSDL document as the messages

are not structured with XML. Within $^{my}$Grid, we have chosen to deal with this by assuming that the information passed around by $^{my}$Grid middleware will be largely opaque to it.

While these two features may appear to be problematic, in practice they simplify the task and scope of the form of semantic service discovery required. As the user is fully involved in the process of service selection, we only require descriptions which reduce the problem space from selection from 1000 services to approximately 10 from which the user can then choose. Other authors have previously noted that this semi-automatic approach simplifies the task of service composition [13], although they see this as a step toward further automation, rather than a strong user requirement. The opacity of the messaging structures means that descriptions do not have to relate to the internal structuring of the data; the best that we can do is describe the existence of a particular datatype[1]. However, the absence of formal structuring means that tools such as WSDL2OWLS [12] are of little use; there is little information in the WSDL file which can be mined from it.

## 4 Coping with Web Service Styles

Of the 1000+ available external bioinformatics services available to $^{my}$Grid users, less than 5% would be considered *plain* web services. Other services consist of approximately:

**25% Soaplab services** Soaplab uses web services, but exposes a stateful CORBA-like interface described later in this section.

**30% Bio-Moby services** The Bio-Moby project provides a registry and messaging format[2] for bioinformatics services [17]. This is not described further, but again, imposes additional semantics over normal web service invocation.

**30% Web based REST services** The Seqhound [10] sequence retrieval system delivers its services through a Representational State Transfer (REST) style interface, where all the information that is required for the service invocation is encoded in a single HTTP GET or POST request.

**10% workflows** $^{my}$Grid allows the incorporation of workflows into larger workflows.

In addition, although not strictly services, local Java applications and Java scripts[3] can also be distributed for use within workflows and so need to be discovered and integrated.

[1] Our analogy here is with MIME types. `text/html` tells you little about the datatype, but is useful none the less

[2] As described early, messaging formats are opaque in bioinformatics, because most of the data is informally structured. Like $^{my}$Grid, Bio-Moby's messaging format reflects this, consisting of a thin enveloped which simply describes the existence of a given datatype

[3] implemented using the BeanShell (`http://www.beanshell.org`)

To illustrate further the additional semantics typically found, in Table 1, we give an example of two different presentations of the same service, in this case a BLAST search. This is one of the most widely used tools within bioinformatics. It uses a DNA or protein sequence to search for similar sequences from a database. As such, it takes a sequence as its main input, along with some of a large set of other parameters which modify the search functionality. One standard presentation of this service, which we describe as a "Document Style" approach, has a single operation which takes a sequence as an input parameter and returns a complex BLAST report which is the standard output of this tool. The second provides a much more "Object Style" interface which requires multiple interactions with the service to perform a single BLAST search. This style of interaction is typified by Soaplab.

| | |
|---|---|
| Document Style | BlastReport performBlast( Sequence, gap, etc. . . ); |
| Object Style | ObjectIdentifier getInstance(); |
| | void setSequence( ObjectIdentifier, Sequence ); |
| | void setGap( ObjectIdentifier, Gap ); |
| | . . . |
| | BlastReport invoke( ObjectIdentifier ); |

**Table 1.** Two different service interfaces to BLAST, a widely used bioinformatics tool. BLAST operates over a biological sequence, has a number of parameters and returns a single complex BLAST report. The "Document Style" interface has a single method taking a complex set of parameters, while the "Object Style" interface uses object identifiers to provide an *ad hoc* object orientation.

To describe the object style of service using, for example, OWL-S would require the use of preconditions (`getInstance` must be called before `setSequence`) and effects (`getInstance` uses resources on the server).

Within $^{my}$Grid, however, we have taken an alternative approach. While there are different web service styles, they are only a limited number. Instead of trying to cope with these service styles through the application of semantic descriptions, we have, instead, used an extensible workflow enactment environment, called FreeFluo [11]. The interaction with the web service is handled through a Java interface or *processor*. While this framework is not fully generic, it appears to answer our requirements; support for a new style of service can be added rapidly–in minutes or hours depending on their service complexity. Moreover, the use of widely adopted language such as Java makes this process considerably cheaper than the use of OWL. In effect, we bury the invocation problem by taking advantage of limited families of service patterns with idiomatic patterns of invocation.

The FreeFluo engine then presents a common abstraction over the individual service styles which is used by both the semantic service discovery component, Feta, and the workflow building environment, Taverna. It is this abstraction which we seek to describe as it is this that the users wish to discover.

## 5 $^{my}$Grid's Data Model of Services

In this section we describe the core data model which we use to describe services. Given the constraints that bioinformatics presents and the support that other parts of the $^{my}$Grid architecture provide for the invocation of services, the key differences between this model and that present within OWL-S are those of omission; we have nothing in this model equivalent to either the grounding or process models and only a subset of the service profile. They are also a few additional features which model the ideas users have about services, but which do not map to the underlying middleware layer.

The majority of the information in the data model was captured in the $^{my}$Grid service ontology described previously [19]. This ontology contains substantial information describing the bioinformatics domain, which acts as an annotation vocabulary including: descriptions of the core bioinformatics data types (e.g. `DNA_sequence`), a characterization of the tasks commonly performed (e.g. `Protein_Analysis`) and a description of the biological entities being investigated (e.g. `homologue`). The core data model is shown as a Conceptual UML class diagram in Figure 1.

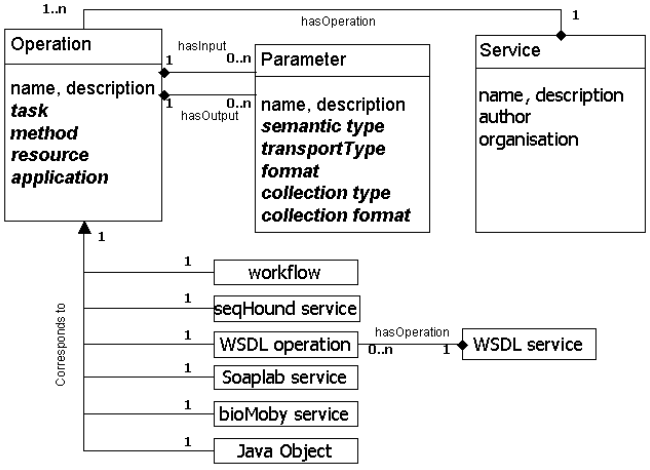


**Fig. 1.** Feta's Data Model of Services: Those attributes filled with terms from the $^{my}$Grid ontology are marked italicized.

Within this data model we distinguish between the core unit of functionality, i.e. the `operation`, and the unit of publication, i.e. the `service`. Our initial analysis of the bioinformatics web services suggested that, in most cases, a Service presented a set of operations providing related but independent functionality. For

this reason, the `service` entity encapsulates information only relating to publication; this includes information such as the provider organization name, the author of the service description, and a free text description of the functionality.

In general, a service may provide one or more service operations. Conventional web services with no state are good examples of this. These operations do not map directly to operations at the WSDL layer. For "object style" services, described in the previous section, the multiple WSDL operations all provide a single unit of functionality from the users perspective. Soaplab services, therefore, are all modelled as a service with a single operation. For other service styles, such as $^{my}$Grid workflows, or Seqhound services, there is no underlying WSDL representation to map to. Again, FreeFluo removes the difficulty of linking between the abstracted service descriptions and these different invocation layers.

The capabilities of `operations`, within Feta, are characterized by the inputs, outputs and several domain specific attributes. All of these attributes use, as fillers, concepts from the $^{my}$Grid domain ontology. These attributes are:

- The overall *task* being performed by the operation. While this attribute has no semantics in the invocation layer, it is an useful description for users who understand the biological *in silico* experiment being performed.
- The underlying *method* being used. Many key bioinformatics tasks can be achieved using more than one algorithm; biologists differ in their level of trust for these different methods, so describing these provides a useful criterion for service selection.
- The *application* to which the service belongs. For example, many service implementations are provided by the EMBOSS project. Again, biologists different in their trust for these different implementations.
- The *resource* that the service uses. Many tools can operate over different data sets. Services providing access to these tools are likely to provide similar or identical invocation interfaces. This attribute enables the biologist to distinguish between these services.

The inputs and outputs of an operation are modelled through the `Parameter` entity. In addition to its name and textual description, a parameter is described with the following attributes:

- The *semantic type* describes the domain specific data type in question, such as `DNA_sequence`.
- The *format* describes the representation of the data. Many data types can be represented using multiple different formats; some services are agnostic to these formats while some are highly specific.
- The *collectionType* and *collectionFormat* attributes are useful where services return a set of results rather than a single item.
- The *configurationParameter* describes whether the parameter is the "main" input or not. This distinction is common in bioinformatics and can best be described by analogy to a unix command line[4]: each command has standard

[4] Biologists normally use the term *parameter* specifically to refer to what we call configuration parameters

input and a set of switches. In general, such parameters have "sensible" default values and can be ignored during service discovery. For this reason, its representation in service descriptions is essential.

## 6 The Feta System Architecture

In this section, we give an architectural overview of the Feta discovery system, as shown in Figure 2. The key characteristic of this architecture is its relative simplicity: the core components communicate through web services; service descriptions are developed using XML and by applying generic XML tooling; querying is performed with Jena using only RDF(S) entailment rather than DL reasoning. Feta is meant as a light-weight semantic search engine rather than a full service registry, so this functionality is deferred to the standard web services registry, namely UDDI [3]. The core components can be grouped as semantic service publishing components (dark grey in Figure 2), service querying components (light grey) and the $^{my}$Grid service ontology (unshaded) used by both.

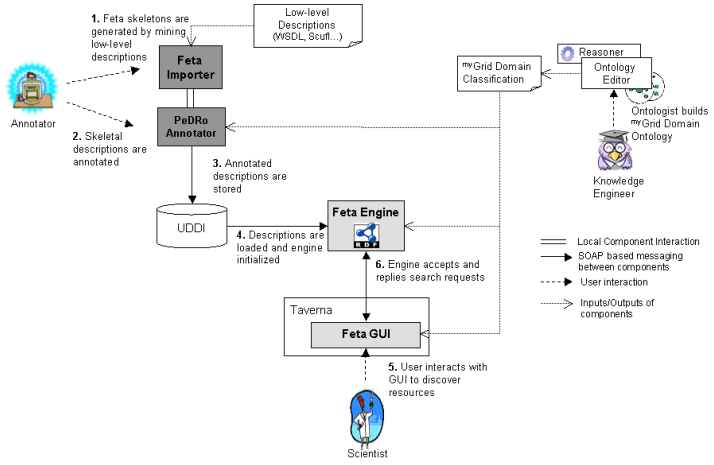


**Fig. 2.** Architectural Overview of Feta

### 6.1 Semantic Service Publication

The requirement for Feta to discover multiple styles of service and to reflect the workflow building scientist's perspective demands manual annotation of the service descriptions. Given that a service's capability is not reflected by its invocation interface provided at the primary publishing stage, it becomes essential for

a secondary publishing stage to take place. It is during publication that the mapping from low-level descriptions of services to the more abstract, user-oriented descriptions takes place.

The absence of formal structuring for most bioinformatics data types (see Section 3), mean that the information which can be obtained from the services themselves is limited to: 1) The Service Name 2) The names and number of service operations. 3) The names and number of operation parameters. For plain web services this information is *imported* from their WSDL files via an XML transformation process. As with the FreeFluo engine, we support an extensibility layer to enable the import of other service styles. In the case of Soaplab services this is achieved by introspective invocation of the service. Other service styles provide information in their own manner and require their own import functionality. In each case, the end product is an XML document conforming to the data model describing in Figure 1, recast as an XML schema. As these documents contain the basic structure for the semantic service descriptions, but little of the information required, we describe them as *skeletons*.

### 6.2 Service Annotation

Following the generation of skeleton documents, manual annotation of these documents is required to provide full descriptions. This annotation process can take considerable time. In the first instance most descriptions have been developed by expert bioinformaticians from within the $^{my}$Grid project. In our experience, the key difficulty has been poor documentation of the services, requiring experimental invocation of the service with test data. More recent experience with service publishing frameworks such as Soaplab, provide documentation directly associated with services which eases this process considerably.

It is clear that tool support is required for this process to encourage either external service providers, or service consumers to generate their own semantic service descriptions. To this end, we use the PeDRo application [6]. This provides a GUI based interface which allows users to generate XML instance documents conformant to a given XML schema. The tool is also ontology aware and can provide easy access to the vocabulary at the point of use. Annotation is limited to named classes rather than fuller class expressions. In Figure 3, we show this application in the process of annotating a plain web service. The tool is not restrictive in the data that is required for the annotator; it is possible to generate descriptions with minimal information to be augmented at a later date as required.

The use of XML, at this point, provides two key advantages: 1) The use of XML schema validation ensures that Feta documents are internally consistent relieving the need for further error checking at later states of the discovery process. 2) Some service providers will already have access to metadata which can be mapped into the Feta schema and used for service discovery. The use of familiar technology should ease the process should service providers choose to bypass the manual annotation step.

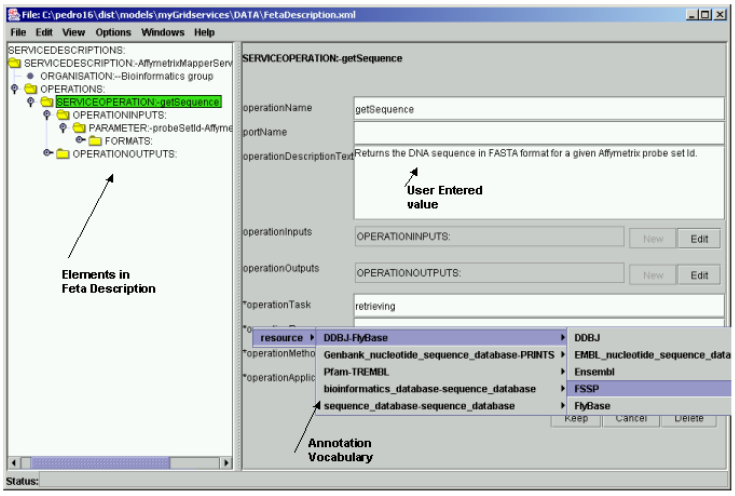


**Fig. 3.** A screen shot of the XML Data Entry Tool PeDRo

The XML document produced is conformant to the data model, described in Figure 1, containing concepts (represented using URL's) to the ontology described in Section 6.3. Currently, the complexity and time-consuming nature of the annotation phase is one of the key reasons why we do not use complex, OWL-based, service descriptions. Annotation providers are generally not conversant with the use of such technology and are unlikely to make use of the expressive power of OWL. Our experience suggest that even the application of a vocabulary is a demanding process.

### 6.3 The $^{my}$Grid domain ontology

It is clear that the domain ontology is a critical component in ensuring the utility of any semantic service discovery architecture. We have discussed previously the approaches of different projects from within bioinformatics for the development of such an ontology [7]. $^{my}$Grid's approach has been to develop a seed ontology which will both provide enough utility for initial users of the system. In turn, this should encourage those in the domain to contribute new terms. This process has previously been used highly successfully in bioinformatics [2].

Due to the complexity of the domain, we choose to develop a complex property based ontology using OWL (initially DAML+OIL), which enabled us to take advantage of the reasoning at development time [19]. For use within Feta, we have reasoned over the ontology and then exported it as an RDF(S) hierarchy.

### 6.4 Querying Feta descriptions

Following the annotation phase, Feta descriptions are published, making use of a UDDI registry. The Feta Engine engine then imports these descriptions, along with the RDF(S) version of the domain ontology, from where they can be queried. The decision to avoid the use of OWL and reasoning technologies at query time enables considerable architectural simplicity at this point. The Feta Engine is essentially a set of canned RDQL queries accessible via a web services interface. We currently use Jena [5] as our implementation backend as its query engine provides support for RDF(S) entailment. The canned queries that we currently support include:

- An operation that accepts input of a given semantic type or something more general.
- An operation that produces output of given semantic type or something more specific.
- An operation that performs a given task (or uses method or uses resource or is part of Application) or something more specific.
- An operation that is of type "WSDL based Web Service Operation", "Soaplab Service", "Scufl Workflow" etc.
- An operation whose name/description contains a given phrase.

### 6.5 The Feta GUI query tool

The focus of semantic discovery in this paper has been to provide support the workflow building. It is clear, therefore, that the discovery architecture needs to be accessed from within Taverna, our workflow building environment. For this purpose, we provide a plug-in which is shown in Figure 4. The query interface enables the user to build a composite search query using the supported canned queries.

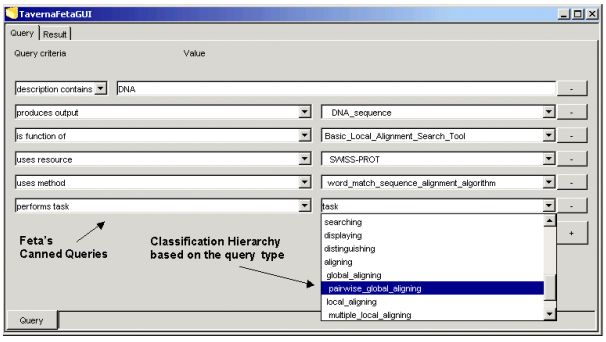


**Fig. 4.** GUI Panel for Building Search Requests

Results of the search are then returned to the user in a results panel shown in Figure 5. Any additional information available about the service is also displayed enabling the user to make the final selection of the most appropriate service. These services can then be added to the workflow by means of drag and drop. Currently returned results are not ranked as most queries narrow the total number of services from which the user can then select manually.

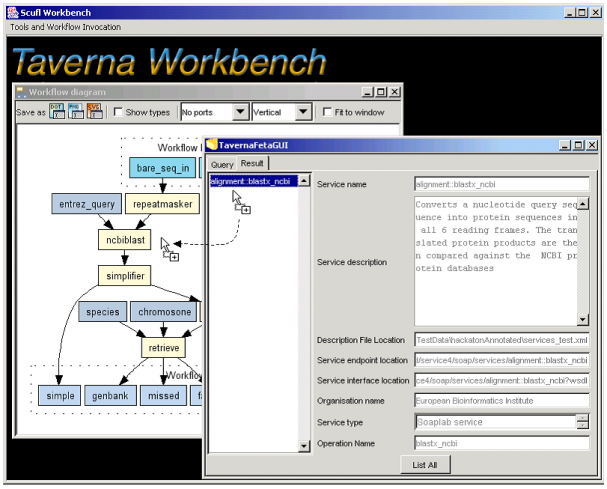


**Fig. 5.** GUI Panel for Displaying Search Results

We currently consider the query interfaces to be preliminary. Although more expert bioinformaticians are comfortable with this kind of boolean query interface, many biologists are not. We would like to pursue further integration within Taverna to alleviate this need. In particular, we pursue the use of workflow context to filter services. Hiding the explicit use of semantic service discovery should enable it to become a more natural part of the process of workflow building.

## 7 Discussion

In this paper, we have described our application of Semantic Web technologies to service discovery within bioinformatics. On the whole, the distinctive features of our system are: 1) Its light-weight semantic support 2) Its semi-automatic approach to discovery 3) Its user-oriented capability-based model that enables discovery.

Within Feta we have adopted light-weight semantics, using an RDF(S) classification and entailment, as this appears to be sufficient in this domain. There is an increasing urgent demand for a publicly available registry and associated search facilities. Our choice of an RDF(S) backend has enabled development and

deployment of a system with low complexity, without precluding migration to a richer semantic framework based on OWL when required. Currently, we have found that reasoning technologies are useful during construction of a domain ontology [19]. However our assessment is that there is a limited role for reasoning technologies in enabling user oriented service discovery. The level of expressivity that the system can cope with is limited not by the complexity of the reasoning task but by the requirements of the biologist end users who develop both the service advertisements and discovery queries, and the capabilities of the user interfaces that are used in their generation.

It is possible that satisfiability checking would prove useful during the generation of service descriptions, as the ontology could be used to express constraints on the use of the vocabulary over and above those already provided by the PeDRo user interface. For example, services descriptions with `DNA_Sequence` inputs and outputs, but performing a `Protein_Analysis` task are likely to be erroneous. However, this sort of constraint checking would require a substantial extension to the ontology. It would also require significant ontological engineering knowledge from the curators; this is likely to discourage the community involvement vital to the development of an accurate representation of the domain [2].

Although we have concentrated on providing appropriate user interfaces and tool support for the process of semantic service description and discovery, there are still some areas of weakness. The $^{my}$Grid ontology, in particular, is a key component of the architecture; without an appropriate model of the bioinformatics, we will not be able to provide appropriate service discovery. There is currently no way for service providers and consumers to provide feedback on this model at the point of use. The Bio-Moby project [17] has a more open and collaborative approach to ontology building, but lacks the quality control that $^{my}$Grid's curatorial approach provides. Better tooling should enable us to take advantage of the best features of both these approaches.

The Semantic Web has always been envisaged to have levels of expressivity– as typified by the Semantic Web Layer Cake [4]. Much of the work on semantic web services has focused on the upper levels of the expressivity. In common with other authors [9], we have found that "being light-weight and flexible trumps other features". We believe that our XML and RDF(S) based architecture fulfils most of the requirements of the bioinformatics domain while retaining the simplicity, which enables us to adapt service discovery to the specific nature of bioinformatics. We suspect that the use of such light-weight architectures with appropriate data models are likely to be very useful in many other domains.

## References


1. S. Arroyo, M. Stollberg, and Y. Ding. WSMO Primer. DERI Working Draft v01, 2004.
2. M. Bada, R. Stevens, C. Goble, Y. Gil, M. Ashburner, J. A. Blake, J. M. Cherry, M. Harris, and S. Lewis. A Short Study on the Success of the Gene Ontology. Accepted for publication in the Journal of Web Semantics, 2004.

3. T. Bellwood. UDDI Version 2.04 API Specification. UDDI Committee Specification, OASIS, July 2002.
4. T. Berners-Lee. Semantic web. XML2000, 2000. `http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html`.
5. J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the Semantic Web Recommendations. Technical report, HP Labs, December 24th 2003.
6. K. L. Garwood, C. F. Taylor, K. J. Runte, A. Brass, S. G. Oliver, and N. W. Paton. Pedro: a configurable data entry tool for XML. *Bioinformatics*, page bth251, 2004.
7. P. Lord, S. Bechhofer, M. D. Wilkinson, G. Schiltz, D. Gessler, D. Hull, C. Goble, and L. Stein. Applying semantic web services to bioinformatics: Experiences gained, lessons learnt. In *International Semantic Web Conference*, pages 350–364, 2004.
8. D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. Bringing Semantics to Web Services: The OWL-S Approach. In *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, Lecture Notes in Computer Science. Springer, July 2004.
9. R. Masuoka, B. Parsia, and Y. Labrou. Task computing - the semantic web meets pervasive computing -. In *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, Sanibel Island, Florida, October 2003.
10. K. Michalickova, G. D. Bader, M. Dumontier, H. Lien, D. B. R. Isserlin, and C. W. Hogue. SeqHound: biological sequence and structure database as a platform for bioinformatics research. *BMC Bioinformatics*, 3(32), 2002.
11. T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.
12. M. Paolucci, N. Srinivasan, K. Sycara, and T. Nishimura. Towards a Semantic Choreography of Web Services: from WSDL to DAML-S. In *In Proceedings of the International Conference on Web Services (ICWS 2003)*, pages 22–26, 2003.
13. E. Sirin, J. Hendler, and B. Parsia. Semi-automatic composition of web services using semantic descriptions. In *Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003*, Angers, France, April 2003.
14. R. Stevens, C. Goble, P. Baker, and A. Brass. A Classification of Tasks in Bioinformatics. *Bioinformatics*, 17(2):180–188, 2001.
15. R. Stevens, H. Tipney, C. Wroe, T. Oinn, M. Senger, P. Lord, C. Goble, A. Brass, and M. Tassabehji. Exploring Williams Beuren Syndrome Using $^{my}$Grid. In *Bioinformatics*, volume 20, pages i303–310, 2004. Intelligent Systems for Molecular Biology (ISMB) 2004.
16. D. Tran, C. Dubay, P. Gorman, and W. Hersh. Applying task analysis to describe and facilitate bioinformatics tasks. *Medinfo*, 2004:818–22, 2004.
17. M. D. Wilkinson, D. Gessler, A. Farmer, and L. Stein. The BioMOBY Project Explores Open-Source, Simple, Extensible Protocols for Enabling Biological Database Interoperability. *Proc Virt Conf Genom and Bioinf*, 3:16–26, 2003.
18. C. Wroe, C. Goble, M. Greenwood, P. Lord, S. Miles, J. Papay, T. Payne, and L. Moreau. Automating experiments using semantic data on a bioinformatics grid. *IEEE Intelligent Systems*, 19(1):48–55, 2004.
19. C. Wroe, R. Stevens, C. Goble, A. Roberts, and M. Greenwood. A Suite of DAML+OIL Ontologies to Describe Bioinformatics Web Services and Data. *The International Journal of Cooperative Information Systems*, 12(2):597–624, 2003.