A SUITE OF DAML+OIL ONTOLOGIES TO

DESCRIBE BIOINFORMATICS WEB SERVICES AND DATA

Chris Wroe, Robert Stevens, Carole Goble, Angus Roberts, Mark Greenwood

Department of Computer Science and [2]School of Biological Sciences,
University of Manchester, Oxford Road, Manchester, M13 9PL
{wroec, robert.stevens, carole, angus, greenwood} @cs.man.ac.uk

http://www.mygrid.org.uk

Tel: +(44)161 275 6239

Fax: +(44)161 275 6204

Abstract

The growing quantity and distribution of bioinformatics resources means that finding and utilising them requires a great deal of expert knowledge, especially as many resources need to be tied together into a workflow to accomplish a useful goal. We want to formally capture at least some of this knowledge within a virtual workbench and middleware framework assist a wider range of biologists in utilising these resources. Different activities require different representations of knowledge. Finding or substituting a service within a workflow is often best supported by a classification. Marshalling and configuring services is best accomplished using a formal description. Both representations are highly interdependent and maintaining consistency between the two by hand is difficult. We report on detail a description logic approach using the web ontology language DAML+OIL that uses property based service descriptions. The ontology is founded on DAML-S to dynamically create service classifications. These classifications are then used to support semantic service matching and discovery in a large grid based middleware project [my]Grid. We describe the extensions necessary to DAML-S in order to support bioinformatics service description; the utility of DAML+OIL in creating dynamic classifications based on formal descriptions; and the implementation of a DAML+OIL ontology service to support partial user-driven service matching and composition.

Introduction

The bioinformatics community exists to apply computational techniques to vast data and knowledge resources in order to answer complex biological questions. The resources are widely distributed, highly heterogeneous, highly diverse and highly autonomous [1]. Traditionally, a bioinformatician takes, say, a single biological sequence, a DNA fragment or a protein, submits this to Web based tools, interprets the results and moves on to the next appropriate analysis. Thus, a characteristic of bioinformatics is the discovery of suitable resources and the marshalling of those resources to work together to perform a task. However, the "craft-based" practice of a biologist undertaking the discovery, interoperation and management of the resources by hand is unsupportable. High throughput experiments such as gene expression arrays now produce 50,000 data points per experiment not 200 and over 500 public databases and applications are available to a working biologist. The increase in data, the proliferation of analysis tools, and the range of knowledge required to interpret and use them requires that the orchestration of resources must be at least partially automated.

The Grid is a distributed computational infrastructure that enables rapid assembly and disassembly of services into transient confederations that accomplish a task wider than that enabled by the individual components [2]. Initially the Grid focused on orchestrating computational resources for expensive and complex analysis for example data mining and simulations in Particle Physics. Now the remit is broader –

the on-demand seamless and dynamic assembly of resources of any kind (sensors, databases, applications, people) into problem solving federations.

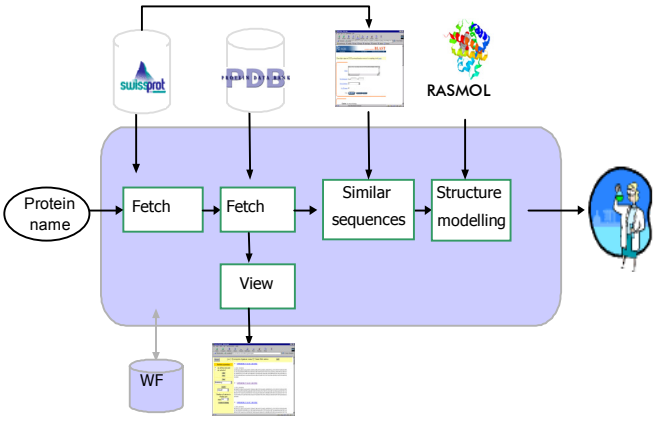

Figure 1. An *in silico* experiment as a workflow

When characterising a protein, a biologist will sometimes wish to investigate the three-dimensional structure of that protein.

1. The biologist has a protein name. She uses this to fetch the matching protein from the SWISS-PROT protein sequence database.
2. She uses the SWISS-PROT cross-link to PDB protein structure database to fetch known three-dimensional structures.
3. Having retrieved and examined the structural data, she might wish to investigate whether other sequences existed with similar structures. As sequence is thought to determine structure, she uses the sequence similarity tool BLAST to find sequences that are sufficiently similar to expect three-dimensional structural similarity.
4. A structure prediction and visualisation tool RASMOL is used to predict and view possible structures for the similar sequences collected.

The workflow can branch to include a viewing task for the PDB structural information retrieved during the workflow.

myGrid[1] is a large UK eScience pilot project to provide open source high-level Grid middleware to enable a virtual workbench for data-intensive bioinformatics. The emphasis is on data integration, workflow, personalisation and provenance. In myGrid resources are services that can be statically or dynamically combined within a context: for example the specific user, the cost of execution, the speed of execution, reliability, the appropriate authorisations available to the user and so on. Service orchestration can take many forms, but in myGrid chiefly falls into two categories[2]:

- Database integration: dynamic distributed query processing, or the creation of virtual databases through federations in the style of TAMBIS [1] or DiscoveryLink[3];
- Workflow orchestration: Process flows, or workflows coordinating and chaining services using a systematic plan, are the manifestation of *in silico* experiments, allowing us to capture and explicitly represent the e-Scientist's experimental process (Figure 1). Workflow is not just document flow.

It is this workflow view of bioinformatics tasks that we specifically take in this paper. An expert bioinformatician knows which services to use; how to use those services; in which order to use those services; how to move data between services and how to reconcile the semantics of those services. A similar knowledge component will be necessary for a machine based eScience infrastructure. Specifically, two key knowledge-rich services are required:

---

[1] http://www.mygrid.org.uk. *The project is in its early stages and common with Grid practice, operates as a series of experimental prototypes.*

[2] *The line between the two categories is not clean cut, and is usually determined by granularity of process. For example, workflow links database integration processes and a virtual database query might be implemented through a query plan executed as a workflow. Such subtlies are not the focus of our discussion,*

- Finding the *type* of services appropriate to a particular task component in a distributed environment (e.g. sequence alignment) and then appropriate *instances* of that service (e.g. BLAST, which is an algorithm for sequence alignment, as delivered by NCBI[3] as an application);
- The appropriate assembly of these discovered services to form transient, dynamic confederations that fulfil the task. By dynamic we mean that services are found, selected and executed while the workflow is being enacted.

This paper seeks to explore the use of ontologies to describe, discover and compose services in a bioinformatics setting. We report on our experiences of building an early myGrid prototype (Version 0). We build on work from the DARPA Distributed Agent Mark-up Language Programme by using the DAML+OIL ontology language [5] and extending the DAML-S service ontology [6]. Our contribution is in extending this work and implementing a *real solution* that operates in a *complex, dynamic bioinformatics middleware project*. We show that DAML+OIL provides an effective language to describe the functionality of service in such a demanding arena, but that, although a useful starting point, DAML-S is so generic it requires considerable extension to be used in practice.

The rest of the paper is organised as follows: Section 2 examines the need for service descriptions, service classifications and constraints on the composition of services. These three aspects are intimately linked and drive the choice of technologies and representations for service metadata. Section 3 examines how the web ontology language DAML+OIL can support the definition of all three aspects and their linkage. Section 4 begins by describing how DAML-S has been extended in myGrid with the use of a suite of ontologies to support bioinformatics service description. It goes on to report on the deployment of these ontologies and associated reasoning framework within the myGrid project. The paper concludes with a discussion of the utility of DAML-S and issues raised using this approach.

1. The description and classification of bioinformatics resources using ontologies.

Finding the right service depends on knowledge of each service. A user will typically have in mind a task they want to perform on a particular kind of data. They must match this task against available services taking into account the function of the service, the data it accepts and produces and the resources it uses to accomplish its goal. Secondly, they must select, from the candidates that can fulfill their task, the one that is best able to achieve the result within the require constraints. This choice depends on metadata concerning function, cost, quality of service, geographical location, and who published it. In much the same way that the Yellow Pages™ are used to find the phone number of a business service, a *service directory* catalogues available web services. Classification of services based on the functionality they provide has been widely adopted by diverse communities as an efficient way of finding suitable services. For example, the EMBOSS suite of bioinformatics applications and repositories has a coarse classification of the 200 or so tools it contains, and free text documentation for each tool. The bioinformatics integration platforms ISYS [11] and BioMOBY [13] use taxonomies for classifying services. The Universal Description, Discovery, and Integration specification (UDDI) [10] supports web service discovery by using a service classification such as UNSPSC [9] or RosettaNet [8]. The Metadata Directory Service (MDS) [2] and Metadata Catalog (MCAT) [12] resource directory frameworks from the Grid community define the properties that can be used to query a resource.

myGrid groups service descriptions into two categories:

- The **domain metadata** covering the domain coverage of the service, or its function. For example, BLASTn is a tool for computing sequence homology that uses the BLAST algorithm over nucleotides;
- The **business metadata**, covering data quality, quality of service, cost, geographical location, authorisation, provenance of data and so on. For example, the BLASTn service offered by the European Bioinformatics Institute is 80% reliable.

[3] *NCBI: National Center for BioInformatics*

myGrid adopts a four-tiered model of services, extending the three-layer model proposed by the Open Grid Services Architecture [15]:

1. **Class of service**: e.g. a protein sequence alignment application, or a protein sequence database. The service descriptions are exclusively described by the domain metadata.
2. **Specific example of an abstract service**: e.g. BLAST or SWISS-PROT. This requires metadata such as the type of inputs and outputs, the characteristics of an algorithm for a computational tool, or the kind of queries that a data repository accepts. The service descriptions are exclusively described by the domain metadata.
3. **Instance service description of a specific service**: e.g. BLAST or SWISS-PROT as offered by the European Bioinformatics Institute (EBI). Many sites offer mirrors or versions of the same service. The service descriptions are exclusively described by the business metadata. We record the version of the service, access cost, authorization permissions and so on;
4. **Invoked instance service description**: e.g. BLAST as offered by the EBI on a particular date, with particular parameters when a service was actually enacted. An important aspect of scientific research, as it is in many other disciplines, is to keep adequate records of how results were obtained. The workflow acts as a template to record this information. For example the provenance metadata will record the time and date of each operation performed in the workflow, how long it took, where it was run and a link to the data produced. Results from each individual run could influence metadata in the respective service's profile such as quality of service. The service descriptions are exclusively described by the business metadata.

In myGrid we seek services initially by their scientific metadata and use the business metadata to choose between services of equivalent scientific functionality. Candidate services may not necessarily be an exact match. We need to reason over the properties of the service descriptions and their classifications to infer close matches that are substitutable, in a similar way, for example, as Matchmaker [14]. Services can be discovered, matched and selected both *before* the workflow is executed or *dynamically* during its execution. Rather than seeking a specific service to perform the operation, we define the functional *class* of service required. For example, we may plan a workflow to use a SWISS-PROT[4] service without specifying the actual SWISS-PROT service to invoke (for example a local implementation or one managed by the European Bioinformatics Institute). This "late binding" of a service protects the running workflow from transient changes in the availability of specific services. This flexibility requires a fine-grained service classification that captures the scientific function. The workflow enactment engine when confronted with a *class* of service rather than a specific instance, interrogates the service directory to retrieve available candidates. It then chooses a candidate using a predefined selection policy, based on additional quality and cost metadata held in the service directory. The selection may even be made interactively by the user. A detailed example of this process is given in section 4.

Services generate data. This data will could be the input to another service or could be stored in a repository, or both. In a workflow, we need to ensure that the type of the data (for example nucleotide sequence) matches the service's input type. The syntactic encoding of the data in a particular formats has to be matched up, but this is straightforward to overcome with format translation tools. More interesting is the semantic *type* of the data must match: for example, a collection of enzymes is permissible as input to BLASTp as enzymes are a kind of protein and BLASTp takes sets of proteins as an input. To guide the user in choosing appropriate operations on their data and constraining which operation should sensibly follow which, it is important to have access to the semantic *type* of data concerned. Consequently, service descriptions should cover the type of their inputs and outputs, and the type should be carried with the data for future use in new workflows.

[4] *A protein sequence database that has a number of mirror sites throughout the world.*

1.1. Description, classification and constraints of services

It is clear that to discover services and configure workflows services should be described and classified. We believe that the descriptions, the classifications and the constraint management are tightly coupled and should be treated together within a uniform framework. For example, a description must be classified, and may be classified in multiple ways. A description should be self-coherent and consistent with respect to others in the classification. The classification should evolve as the descriptions evolve. The classification should be self-consistent, sound and complete. Figure 2 summaries the inter-related and inter-dependent properties of service metadata.

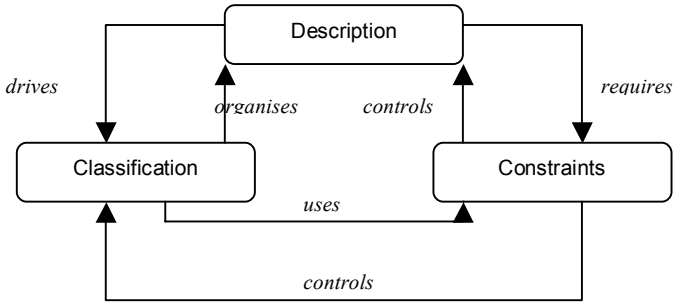

Figure 2: the relationship between service description, classification and constraints

The desirable properties of each aspect are detailed below:

Descriptions should be: (a) sufficiently rich to capture as little or as much of the capability of the service; (b) flexible and extensible as the service and its use evolves; (c) support arbitrary granularity; support alternative interpretations based on context; and (d) should not pre-suppose a complete and comprehensive prescriptive and pre-hoc enumeration of every kind of service capability that will be offered and sought.

Ideally descriptions should be formal and rigorous enough to withstand automated processing by, for example, reasoning engines that check if two descriptions are equivalent, subsume one another or at least partially match, or are mutually inconsistent.

Classifications of the descriptions act as: (a) an index to the descriptions; (b) a filtering mechanism; (c) a query containment mechanism; and (d) a substitution mechanism such that partial or imprecise matches can be catered for and a clustering mechanism for similar services. The classifications are lattices, not trees, as a description will have multiple parents. For example, a protein might be classified by what it transports, what it catalyses, the process it participates in; and where it is located. A service may be classified by its location, its cost, its inputs, its function and so on. There will be many classifications for the same description depending on the view taken. This reflects the diversity of the metadata required to adequately describe the services. Using multiply classifications to organize a description from different viewpoints is sometimes known as "multi-axial classification".

Constraint mechanisms on the descriptions and classifications for controlling configurations of services and guiding service composition. The descriptions are used as a typing mechanism, making the classification a type schema. The key requirement here is that mechanisms are in place to assist in determining whether a combination of descriptions is coherent and valid.

myGrid uses a suite of ontologies to represent these three aspects of metadata. Ontologies provide a *vocabulary* of terms or concepts to form descriptions. Using free text to describe service is expressive and flexible, but difficult to search, to reason over, or to automatically organize into classifications. Controlled vocabularies are better. Ontologies provide at the very least a *taxonomy* that organizes the concepts or terms into a classification structure. Well-known service classifications such as RosettaNet and UNSPSC try to pre-enumerate all possible terms into a single taxonomy. SYS, BioMOBY and EMBOSS use a similar approach. Other relationships and axioms capture and *constrain* the properties of the concept.

Experience from the long history of classification systems in the biomedical domain [29] shows that manually created classifications are inflexible and hard to manage when they become large, detailed and multi-axial. A key problem is that many terms are combinatorial cross-products of orthogonal terms. For example, in ICD10 there are 817 different terms for accidents involving bicycles, produced by hand through the cross-product of five different axis [16]. This phenomenon is prevalent in Life Sciences. For example, the Gene Ontology has adopted an ad-hoc solution for cross products [17]. Another problem is that when the descriptions become detailed they become very long. This is known as the "long label" problem. For example

Consider the simple taxonomy in figure 3 describing sequence alignment services.

(a) The service descriptions are insufficiently comprehensive and hide important properties. For example, the BLASTn description implicitly describes a service that operates over nucleotides (and not proteins). The description "protein pairwise alignment" doesn't exist despite the fact that this is a likely service description.

(b) The descriptions should be classified and hence queried in multiple ways giving multiple classifications. For example, we should classify descriptions by operation (alignment, pairwise, multiple), by data source (protein, nucleotide, sequence), or by algorithm (SmithWaterman, BLAST).

(c) Constraints are missing for the descriptions. BLASTp only operates over proteins. tBLASTn compares a protein sequence query against a nucleotide sequence database dynamically translated in 6 reading frames. Alignment is an operation that only applies to sequences and not pathways, and at least two inputs are required.

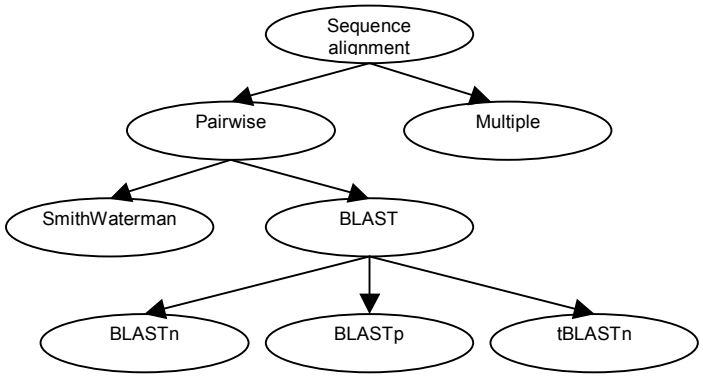

Figure 3: a simple single axial ontology describing sequence alignment services

A richer ontology moves away from a pre-enumerated taxonomy of terms to a classification of concepts based on the *properties* of the concepts. As the properties change so should the classification. Then the classifications accurately reflect the concept descriptions. A mechanism for controlled post-coordination of terms, where new concepts are formed from the combination of others, is highly desirable. This is the approach adopted by SNOMED [18]. The analogy is that, instead of trying to describe a controlled vocabulary by enumerating all known noun phrases, instead determine a core vocabulary and a grammar with rules to form new phrases.

As the manual maintenance of large and evolving classifications is unmanageable, automated support is essential [29,22]. We use a suite of **ontologies** expressed in **DAML+OIL**, to provide: (a) service classifications; (b) a vocabulary for expressing service descriptions and (c) a reasoning process to both manage the coherency of the classifications and the descriptions when they are *created*, and the service discovery, matching and composition when they are *deployed*.

2. DAML+OIL

DAML+OIL is an ontology language specifically designed for the Web. The notion of the Semantic Web, which aims to move from syntactic interoperability to semantic interoperability [19], relies on machine

interpretable semantic descriptions. Thus, DAML+OIL builds upon existing Web standards, such as XML and RDF, and is underpinned by an expressive **Description Logic (DL)** [5]. It is these formal semantics that enable machine interpretation and reasoning support. Formal semantics are necessary for such machine interpretation and additionally aid human communication--an aim of ontological description. DAML+OIL is the basis of the W3C Semantic Web Activity WebOnt standardization process.

DAML+OIL takes an object-oriented approach, with the structure of the domain being described in terms of *classes* and *properties*. An ontology consists of a set of *axioms* that assert, e.g., subsumption relationships between classes or properties. Asserting that services (pairs of services) are instances of DAML+OIL classes (properties) is left to RDF. DAML+OIL also supports the full range of XML Schema datatypes. Overall, it has a model theoretic semantics, and a rich set of constraints available for class descriptions.

From a formal point of view, DAML+OIL is equivalent to a very expressive description logic, with a DAML+OIL ontology corresponding to a DL terminology (Tbox). As in a DL, DAML+OIL classes can be names (URIs) or *expressions*, and a variety of *constructors* are provided for building class expressions. The expressive power of the language is determined by the class (and property) constructors supported, and by the kinds of axiom supported. These axioms make it possible to assert subsumption or equivalence with respect to classes or properties, the disjointness of classes, the equivalence or non-equivalence of individuals (services), and various properties of properties. Classes can be combined using conjunction, disjunction and negation. Within properties both universal and existential quantification are allowed, as well as more exact cardinality constraints. Range and domain restrictions are allowed in the definition of properties, which themselves can be arranged in hierarchies. Properties can also be defined as symmetric, functional or transitive (which is particularly important when representing partonomies).

A crucial feature of DAML+OIL is that subClassOf and sameClassAs axioms can be applied to arbitrary class expressions. This provides greatly increased expressive power with respect to standard frame-based languages where such axioms are invariably restricted to the form where the left hand side is an atomic name, there is only one such axiom per name, and there are no cycles (the class on the right hand side of an axiom cannot refer, either directly or indirectly, to the class name on the left hand side). Thus, classes and class expressions can be used in *descriptions of classes*, as can concrete types such as string and integer. Classes can either be *primitive* or *defined*. Primitive classes are asserted manually into the classification in the style of traditional frame-based modeling. Defined classes are expressions capturing the necessary and sufficient conditions for being a member of a class. Their place in the classification is inferred by merit of their description. Hence the name "description logics"–logics that reason about the descriptions of the class expressions.

All this expressivity may be used in class descriptions, and can be used to capture bioinformatics and molecular biology domain knowledge with high fidelity. However, it is also possible to simply assert class names within a taxonomic structure. Concept definitions can be as simple as possible yet as complex as necessary. Thus DAML+OIL is capable of encoding a full range of ontologies, but its power lies in the possibility of formal description and the reasoning it can then support [22].

2.1. Reasoning in DAML+OIL

DAML+OIL chiefly supports two kinds of reasoning tasks:

(a) The automatic determination of subsumption between compositional descriptions. Given two conceptual definitions A and B, we can determine whether A subsumes B, in other words whether every instance of B is necessarily an instance of A. Consequently, a collection of conceptual definitions can be organised into a multi-axial classification based on the subsumption relation inferred by reasoning about their the definition of the concept expression. Thus defined classes have their position in the lattice determined automatically, and classification is a dynamic process where new compositional expessions can be added to an existing hierarchy. Given a concept definition, we can retrieve all the instances of that concept (which of course includes all instances of subsumed concepts).

(b) A concept satisfiability test on an arbitrary class expression to test for its logical coherency with respect to the concepts in the ontology.

Reasoning can be useful at many stages during the design, maintenance and deployment of large, complex and multiply authored ontologies [5]:

- To support ontology ***development*** by improving the quality of the ontology by checking for logically inconsistent classes and (possibly unexpected) implicit subsumption relationships.
- To support ontology ***deployment.*** The reasoning services are available to any application using the ontology. Concept expressions can be made "on the fly" by the application and classified. This enables, queries to be asked of the ontology, for example "which services use SWISS-PROT as a resource?". By classifying a posed description of a desired service and identifying services that are classified through subsumption reasoning as being more general or more specific. Thus, a service can be proposed as a potential match, substitutable for the one required, without requiring it to be an exact match [14].

3. Describing services in myGrid using DAML+OIL

3.1. Building DAML+OIL ontologies

We use the OilEd graphical editor [25] for developing our ontologies. OilEd uses the FaCT reasoner [20] to support the ontology building process. This style of conceptualisation offers a different development route for the ontology builder. In an extreme case, classes are described and the reasoner used to perform all classification. It is usual, however, to provide a skeleton of asserted concepts and then add properties and the use of reasoning to create a complete ontology.

A. DAML-S acts as a foundation

The DAML-S consortium wish to use a DAML+OIL ontology to provide the framework in which to describe the properties and capabilities of web services [6]. DAML-S provides a mechanism for describing a service's properties and functionality and so is a sensible starting point for developing a descriptive framework for bioinformatics web services. By encapsulating a workflow within a composed service it also provides a mechanism for specifying the structure of a workflow. The service ontology aims to allow:

(a) *Discovery* of an appropriate Web Service within a registry by its properties and capabilities;

(b) *Invocation* by some agent;

(c) *Interoperability* is increased by describing the semantic type of inputs and outputs;

(d) *Composition* of new services;

(e) Verification *of a service's properties;*

(f) *Execution monitoring by tracking what is happening to the described aspects of a service and its sub-services.*

Just as DAML+OIL describes domain knowledge in terms of classes and properties, a DAML-S ontology describes services in terms of their classes and properties. The highest level class is that of `service`. This class of service can be specialised along domain and functional lines; for example `bioinformatics_services`. This upper portion describes general properties of *services* and these properties are divided into three aspects:

(1) Service Profile: Describes what a service requires from an application and what it can offer an application. The *service_profile* is *presented* by a *service*. There is a description of the service and its provider (name, purpose, email, URL, etc.), function behaviour of the service (response guarantee, cost, etc.) and some functional attributes by which the service may be discovered.

(2) Service Model: Describes how a service works – possibly in terms of workflows or execution paths. A *service* is *described* by a *service_model.*

(3) Service Grounding This allows a service invocation message to be built and a response message interpreted. It provides the description of the service's binding to the actual Web Service and reflects some of the information found in a WSDL description of a service.

Here we focus on the tasks of discovery and composition. We postulate that these have most to gain from a DAML+OIL approach because of the need for integrated description, classification and constraints illustrated in section 2. These tasks rely heavily on the information in the service model and certain aspects of the service profile but much less so on the service grounding, which supports enactment and monitoring. We will therefore examine closely the DAML-S service profile and model and describe the extensions necessary to DAML-S in order to support the description of services in the bioinformatics domain.

B. Extending DAML-S in terms of properties

There are a least three major steps in discovering and preparing a service for use.

(1) A set of candidate services is found that will fulfill the necessary task using a classification as a mechanism for indexing and querying the directory of existing services.

(2) A single choice is made based on additional attributes of each service such as cost or quality of service.

(3) The service is configured to perform the specific task in hand.

In the bioinformatics domain, the first step is dependent solely on a functional description of a service and so the first priority and to ensure we can provide adequate functional descriptions of bioinformatics services to support this first 'matching' step. This functional description also provides the information needed to guide service composition.

The DAML-S service profile and model have a set of relevant properties `input`, `output`, `precondition`, and `effect`, which are collectively called `parameters`. We found input and output to be vital descriptive elements in service matching and composition. In addition to inputs and outputs we felt it necessary to add a set of properties to the service profile to capture common ways of describing bioinformatics service, listed below.

- **Task**: The task being performed with the data is an important aspect of a functional description. The task itself is a generic process that should be, in most if not all cases, domain independent. A `performs_task` property was therefore added to the DAML-S profile. `retrieving` is an example of a generic task which would be used with this property.
- **Static information resources**: Many operations especially in bioinformatics use additional informatics resources to complete a task and this forms a major component of a service's functional description. For example, a bioinformatics service may retrieve information from a SWISS-PROT protein database. A `uses_resource` property was therefore added to the DAML-S service model.
- **Existing software tools**: Bioinformaticians will often describe operations using the identity of the tool that implements. The `is_function_of` property was added to allow this description. This is orthogonal to the description of the service that wraps the tool.
- **Software algorithms**: Although the myGrid framework may have no access to the internal structure of many service operations, it is often important to name the algorithm that is used. For example a sorting service could use bubble sort, ripple sort etc. In bioinformatics this is particularly important for sequence alignment and clustering operations. A `performs_task` property was therefore added to the DAML-S profile.

The DAML-S service profile includes many properties such as `geographicRadius`, `providedBy`, `qualityRating`. We found in the bioinformatics domain that this information although important, is secondary and used in the later stages of service discovery and use.

The DAML-S process model describes the operation or operations provided by a service, which are distinguished into atomic, simple, or composite. Atomic service operations can be described in terms of inputs, outputs, preconditions and effects. Composite operation can be described as atomic operations chained together in a workflow. The use of *control constructs* such as *sequence, if-then-else, and fork*, etc,

allows the ordering and the conditional execution of processes in the composition. However, there is no intrinsic support in the DAML+OIL language to define process control or dataflows. Therefore, any information regarding such information cannot be used to calculate a classification structure by reasoners such as FaCT. Also, there are no widely available enactment engines that will interpret the control constructs and orchestrate individual services accordingly. One approach taken by members of the DAML-S consortium is to translate these constructs into an alternative representation (Petri Nets) [23] that is amenable to verification and enactment. Our approach is to represent only minimal composition information within DAML+OIL and to use a workflow language WSFL [26] to represent control structures and dataflows. Enactment engines for WSFL do exist and are used in the myGrid project. Other alternatives from the software industry also exist such as Microsoft's XLANG [27].

We use only one property from the DAML-S service model, `composedOf`. DAML+OIL allows properties to be specified as transitive. We have specified that the `composedOf` property be transitive to aid in classification and query. For example consider a service encapsulating a workflow that includes a step using BLAST. This complex service may in turn be chained together with others to provide a more substantial workflow. The more substantial workflow will still be classified as a service composed of a BLAST operation even though that composition is indirect. In this case the representation of process decomposition informs terminological reasoning by FACT.

C. Extending DAML-S in terms of domain ontologies

The DAML-S service ontology with additions mentioned above provides the basic schema for describing a web service but does not provide the vocabulary with which to describe specific services in the bioinformatics domain for example. We therefore built a suite of DAML+OIL ontologies specific to bioinformatics and molecular biology to allow such services to be described, shown in Figure 4.

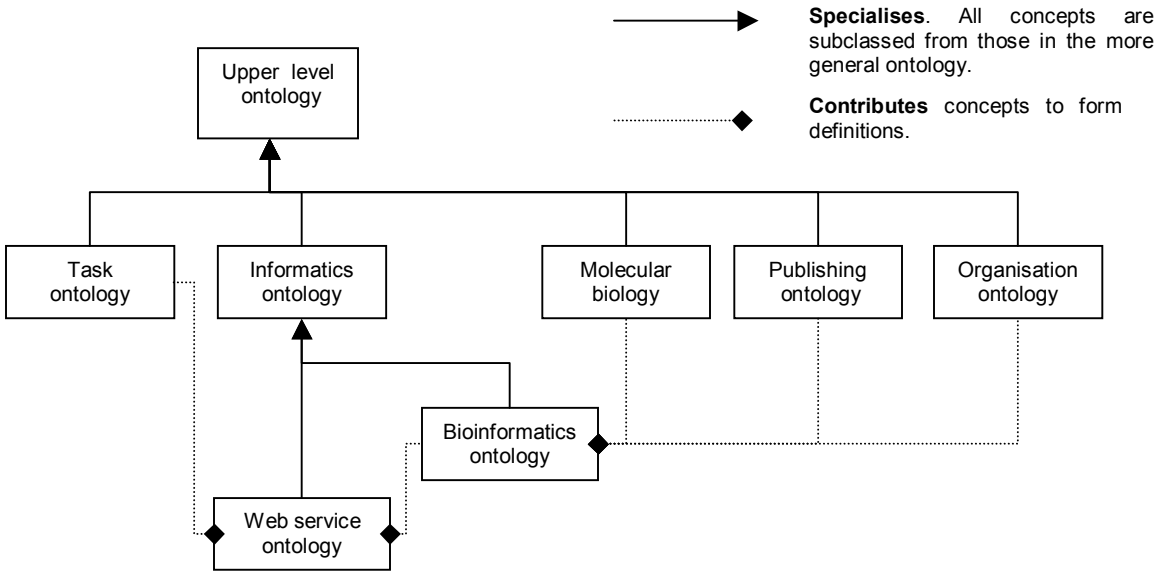

Figure 4. Suite of ontologies used in myGrid and their inter-relationships

- **Standard upper level ontology:** A standard upper ontology forms the foundation for the suite of ontologies, Within myGrid we chose to use the upper ontology based on that used in the large scale medical ontology engineering exercise in the GALEN projects [32]. While not essential, its use does confer a number of benefits:

(a) helps to enforce explicit distinctions between high level categories such as structure and substance.

(b) helps in specifying the constraints in terms of abstract concepts such as modifier, or structure.

(c) helps enforce consistency between ontologies, which is important if descriptions are to be composed from concepts represented in numerous ontologies.

- **Informatics ontology:** An informatics ontology captures the key concepts of data, data structures, databases, metadata and so forth.

- **Bioinformatics ontology:** A bioinformatics ontology builds on the informatics ontology adding specific types of bioinformatics resource such as `SWISS-PROT database`, `BLAST application`, and specific bioinformatics data such as `protein sequence`. By explicitly separating general informatics concepts from more specific concepts applicable only to bioinformatics we hope to reuse as much as possible of the ontology suite for other domains. The majority of bioinformatics data can be characterised by the biological entity it is characterising. The concepts in the bioinformatics ontology commonly reference concepts in the molecular biology ontology.

- **Molecular biology ontology:** Large molecular biological 'ontologies' already exist, such as the Gene Ontology [30], but they are mainly designed to allow annotation of specific data instances rather than specify more abstract *types* of data. The TAMBIS Ontology (TAO)[5][34] is one example of a biological domain ontology used for data integration and as such has many of the higher level concepts required to describe the bioinformatics data types which go on to form the inputs and outputs of services. Examples of concepts include `protein`, `nucleic acid`, and `sequence`.

- **DAML-S web service ontology:** As the DAML-S service ontology is designed specifically to support web services it becomes an extension of the informatics ontology.

- **Publishing ontology:** The majority of accumulated biological knowledge resides in published scientific literature and so a significant proportion of bioinformatics services will become available to access and process this information. A publishing ontology is therefore necessary to provide the vocabulary with which capture the functionality of these services. Examples of concepts include `article`, `abstract`, `citation` and `reference`.

- **Organisation ontology:** Currently a minimal ontology with which to categorise the many instances of bioinformatics organisation that occur in service descriptions as the publishers of services or resources. For example, National Library of Medicine (NLM), National Center for Biotechnology Information (NCBI), European Bionformatics Institute (EBI).

D. Bringing it all together

Obviously, the scope of the ontologies is determined by what they are required to describe. In the early stages of the project, the number of available bioinformatics web services was limited. The content of the ontologies must however greatly exceed the current state. Adding subsequent services must be as easy as possible and not require a large overhead of ontology authoring. The vocabulary should be there to describe what is being published. Secondly, demonstration of the utility of formal ontologies for service retrieval requires a service description haystack in which to find the required needle.

Examples of bioinformatics service operations were gathered from three sources.

(1) A list of services implemented by the European Bioinformatics Institute (EBI), one of the project partners, to be used initially by myGrid.version0 but later published for general use.

(2) Functional descriptions of a sample of the tools available in the EMBOSS package [7].

(3) Hypothetical service functionality extrapolated from that embodied in the web interfaces to existing bioinfomatics tools.

---

[5] *The DAML+OIL version of the TAMBIS ontology in is available at http://img.cs.man.ac.uk/stevens/tambis-oil.html*

Service operations were manually grouped into related categories. In the case of EMBOSS the software suite is supplied with its own categorisation. Each category of operations were authored as a block so, for example, all the sequence alignment operations were authored together, because the definitions for each operation are likely to use related vocabulary. For each service operation, an informal description was manually authored from reference material such as online documentation. This acted as a specification of the information to be formally captured in DAML+OIL. From this we gathered a list of concepts necessary to provide those descriptions. These were then entered into the relevant ontologies and the complete DAML+OIL service operation description written using those concepts. Table 1 shows a group of sequence alignment operations and their informal description. Figure 5 gives an example of the formal definitions for one of the operations. Table 2 shows the vocabulary necessary to support this single service class description and how the origin of the concepts is distributed through a range of supporting ontologies.

Service operation name	**Description**
EMBOSS needle operation	global alignment of sequence data using Needleman and Wunsch algorithm to produce sequence alignment data
EMBOSS stretcher operation	global alignment of sequence data using a subtitution matrix to produce sequence alignment data
EMBOSS water operation	local alignment of sequence data using Smith-Waterman algorithm to produce sequence alignment data
EMBOSS wordmatch operation	local alignment of sequence data using word match algorithm to produce sequence alignment data
ClustalW_operation	multiple local alignment of 3 of more sequences using CLUSTALW program to produce multiple sequence alignment data
BLASTp operation	compares an amino acid query sequence against a protein sequence database
BLASTn operation	compares a nucleotide query sequence against a nucleotide sequence database
tBLASTn operation	compares a protein query sequence against a nucleotide sequence database

Table 1. Sequence alignment operations and their informal description

Concise definition

class-def defined BLAST-n_service_operation
  subclass-of atomic_service_operation
  has_Class *performs_task* pairwise_local_aligning
  has_Class *produces_result* sequence_alignment_report
  has_Class *uses_resource* nucleotide database
  has_Class *requires_input* nucleotide sequence
  has_Class *is_function_of* BLAST_application

Fully expanded definition

class-def defined BLAST-n_service_operation
  subclass-of atomic_service_operation
  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 5. Concise and fully expanded formal description of the BLAST-n service operation written in a human-readable pseudo version of DAML+OIL, in the spirit of the OIL human-readable syntax [22]

Ontology	Concepts
Informatics	report, data, database
Bioinformatics	BLAST_application
DAML-S service	service_operation, atomic_service_operation
Molecular biology	nucleic_acid_molecule, sequence
Task ontology	aligning, pairwise, local

Table 2. Contribution of each ontology to example service description.

At this stage we have concentrated on description of services rather than their classification. Figure 6a shows the classification before we use the reasoner as a completely flat classification with no organization. The FaCT reasoner is used to infer subsumption relationships we see in Figure 6b

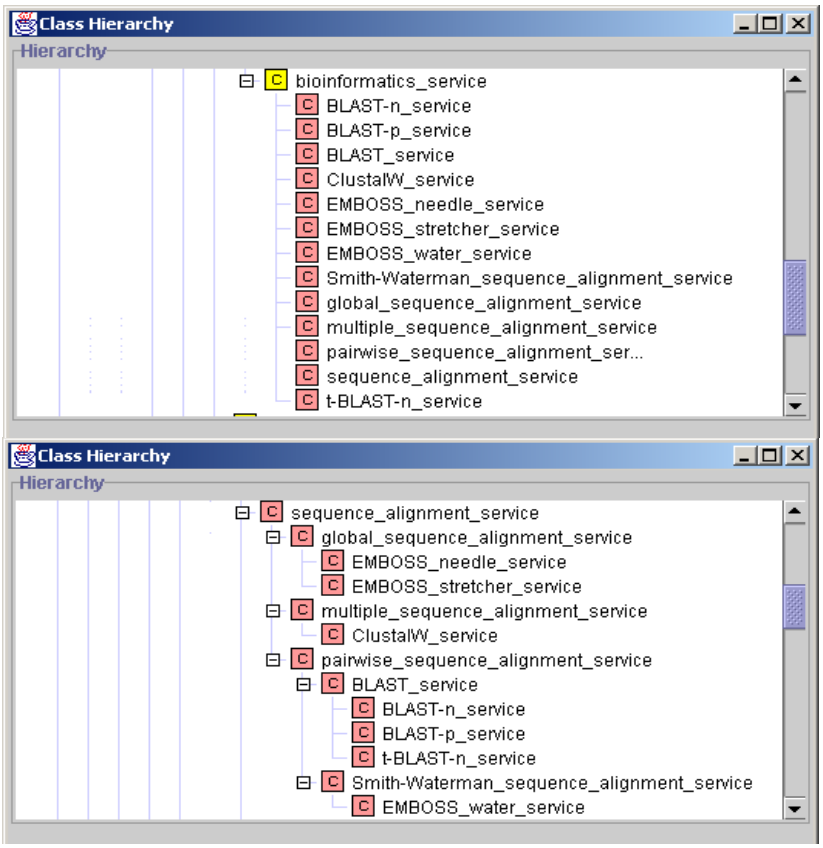

Figure 6. Service classification before (a) and after reasoning has occurred (b).

E. Organisation

The classification is built from the bottom up adding definitions for the most specific classes that capture the functionality of service operations. An additional set of more abstract classes are required to organise the hierarchy to enable the classification to be browsed by users. More abstract service operations are defined by using only a subset of the available properties or by using more general values for properties. An example of such a class is the '`protein sequence alignment operation`' which is designed specifically to accept protein sequences. The human readable version of the definition is shown in figure 7.

class-def defined protein_sequence_alignment_operation

subclass-of service_operation

has_Class *performs_task* aligning

has_Class *produces_result (*report **has_Class** *is_report_of* sequence_alignment)

has_Class *requires_input* (sequence **has_Class** *is_sequence_of* protein_molecule)

Figure 7. Fully expanded formal description of the protein sequence alignment operation written in a human-readable pseudo version of DAML+OIL.

In these early stages of the project, the choice of abstract classes to organise the classification has been arbitrary. In other areas of biomedicine, classifications have long been used to categories such things as species and disease. If one were to present a species classification to a biologist, they would expect it to be organised in according to the Linnean system (the widely adopted classification system for biological species) and probably the variant supplied by the NCBI. However, bioinformatics service classifications are novel with only a few examples existing in integration projects such as ISYS and BioMOBY. As these classifications mature it may become necessary to organise a classification to the emerging norm. The ability to add and remove class definitions at will, without the need to manually maintain the organisation of the hierarchy, allows a DAML+OIL classification to be adapted to whatever hierarchical view is needed. This organisation may be adapted to reflect the experience of the user and the context of the session. A bioinformatician with knowledge of existing tools may prefer to have services classified according to the tools which implement those services. In contrast, a biologist with no previous experience of using such tools may well much prefer to search for services based on the kind of data they have and the task they wish to perform.

F. Coverage

The scope of the ontologies is limited to support service discovery. Each ontology tends to contain abstract concepts only. For example, the main use of the molecular biology ontology is in describing the semantic content of bioinformatic datatypes, such as protein sequence data. There is no need at the moment to describe the thousands of specific classes of proteins that exist. We do however envisage more detailed domain ontologies will be required for other aspects of the myGrid project. One example is in filtering notification events. A key aim of myGrid is to notify the user when data has been updated if that data applies to their particular area of interest, or if it may modify the results or previously run workflows. The filtering mechanism needs to understand not just the type of data but also the content. By separating the ontologies into a suite of independent and reusable units we hope to minimise the effort needed to include existing content focused ontologies such as GO and expand the vocabulary as required. Table 3 shows the number of concepts and properties present in each ontology, and the size of the vocabulary needed to express the concept definitions.

Ontology	N^O of Classes (primitive/ defined)	N^O of Properties	Size of Vocabulary used to form concept descriptions	Individuals
Biology	112 (66/46)	22	-	-
Publishing	6(6/0)	-	-	-
Service	117(1/115)	8	124	-
Informatics	96 (48/48)	7	-	-
Bioinformatics	75 (31/44)	9	-	-
Upper level ontology	50(40/10)	7	-	-
Organisation	1 (1/0)	0	-	8

Table 3: Size of each ontology in terms of DAML+OIL classes and properties, and the size of the vocabulary used to define those classes.

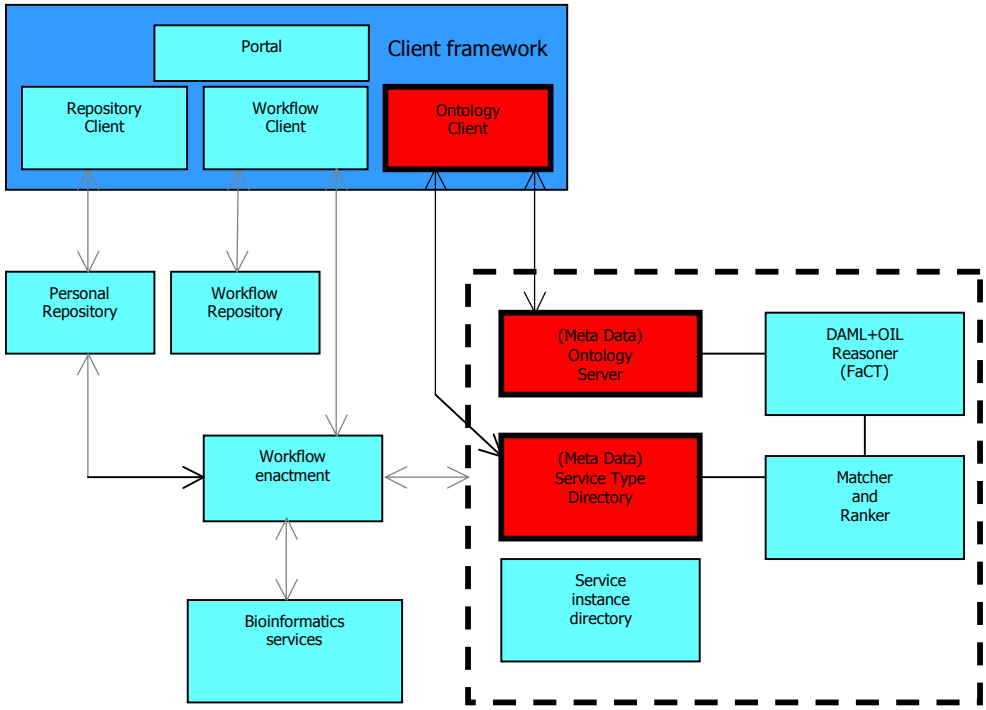

Figure 8. Diagram showing core components of myGrid.version0 architecture. Specifically highlighted are the ontology server, client, and service type directory that allows a user to match their requirements against existing classes of available services.

3.2. Deploying DAML+OIL ontologies

The architecture of myGrid.version0 is given in Figure 8. The dotted line encompasses the components that are relevant to service description and discovery. The core components are:

- A **web portal** provides a user interface component that provides access to the framework by the user.
- A **user repository** is a store of personal data and provenance information. It acts as an "electronic lab book" in which completed results are automatically recorded.
- A **workflow repository** is a store of workflows.
- A **workflow enactment engine** is a service that interprets workflow definitions and then discovers and invokes the necessary domain services on the Grid in the correct sequence.

Key components for service matching and discovery are:

- A **ontology server** provides access to the DAML+OIL data structures and allows the ontology to be submitted to the FaCT reasoner. We use a DAML+OIL ontology server from a related project, Conceptual Open Hypermedia Systems Environment (COHSE)[28]. Additional functionality was

added to the COHSE ontology server to support introspection of the metamodel and that of DAML+OIL class descriptions;

- A **FaCT reasoner** to perform class subsumption and satisfiability reasoning described in section 3;
- A **ontology client** allows users to create DAML+OIL descriptions of their requirements without knowledge of DAML+OIL syntax;
- A **service type directory** makes available the dynamic DAML+OIL service classification, supported in this task by the ontology server;
- A **service directory**, an extended implementation of UDDI, is used to register service instances;
- A **matcher and ranker** used to manage the ranking of candidate descriptions. Similarity metrics are essential if we are to rank results of a service query Approaches such as those describe in Matchmaker[14] use metrics based on distances (or number of links) between terms. This approach works well with static models where the terms are fixed in a topology, but is less appropriate with the dynamic classification of a DL. The "distance" or number of links between terms may vary depending on the compositions that have been constructed. An alternative solution is a model of similarity based on the premise that two terms are similar if they share a number of parents in the concept hierarchy. Metric are defined based on common and distinguishing features. We have implemented such an approach in the past [33].

The architecture in figure 8 has been built around simple scenarios reflecting plausible bioinformatics tasks. The scenarios follow a common theme in bioinformatics, in which a biologist has experimentally acquired some new data and wishes to relate this new data with previously obtained information, in the hope of extending existing knowledge.

A. Functionality in the target application

As in the DAML-S work, we wish to use subsumption provided by the DAML+OIL reasoner to match a users functional requirements with those present in the existing of service classification. In effect the user describes their desired class of service as a DAML+OIL class definition and this is classified using FaCT in relation to what classes already exist. Candidate services are those that are *subsumed by* the user's description.

We also wish to constrain how services can be composed together based on the description of inputs and outputs to each service. If the first service in a workflow returns protein sequence data then the choice of second step should be constrained to services that accept protein sequence data.

B. Integration in the target application

A DAML+OIL approach to service matching requires the user to construct an ontological description of the operation they require using the model with which all service operations have previously been described. We envisage few, if any, of our users will be acquainted with ontology authoring environments. A simplified user interface is required to support such a process. It needs to guide and constrain the user in what they can describe. It must hide the underlying formalism used to represent ontological definitions, instead presenting a simplified form based interface from which the underlying representation can be generated. This requires a user interface to have knowledge of the model used to describe a service operation and the possible values for each attribute with which the operation can be described.

We have implemented a separate ontology that contains the metamodel for describing a service operation. The entries in the metamodel are called 'reasonables' taken from the fact that they describe how it is reasonable to describe or specialise a particular concept [31]. When the web portal is constructing the user interface to describe the users functional requirements, it consults this metamodel to determine the specific drop-down lists to display and the values in those lists. The content of the reasonables store reflects the extended DAML-S service profile and service model described earlier with value sets tailored to the bioinformatic setting.

C. Walkthrough

As myGrid is bringing together a range of web service and metadata technologies it was important that we demonstrate early on that we are producing a cohesive framework. The aim was not to demonstrate the full functionality of all components, but rather to ensure that core components can work in concert to provide basic functionality

Once the core components are running they are populated with information about available domain services. The suite of ontologies are loaded into the ontology server and service type directory. Corresponding instances of these services must be registered in the service directory. These services are implemented and interact successfully with the workflow enactment engine, which is to invoke them.

Following successful deployment, the user can access the functionality of the framework through a web portal. The portal provides access to the user's personal repository. Into this repository the user has placed their new experimental data, in this case a protein sequence. Currently the user manually annotates this data with an ontology concept describing its semantic type.

Once the user has the relevant data in their personal repository, they can proceed to constructing a workflow with which to process that data. The user finds the appropriate service by constructing a description of what that service should do. The description is written by selecting options in a drop down list that cover the major aspects of the desired service operation (see figure 9). Selecting these options result in the construction of a new service operation concept. Not all options have to be filled in. The description can vary in terms of completeness and abstraction. Certain slots may already be constrained by other choices. For example, the semantic type of the data is provided by the user repository and is automatically entered into the required input slot.

Once the user is satisfied they have captured their requirements, the description is matched against the pre-existing service classification. The matching process is achieved automatically by a number of steps. Firstly, the user description is converted into a DAML+OIL concept definition. This definition is submitted to the ontology server. The new partial service operation is added to the service ontology, which is then submitted to the FACT reasoner. The reasoner calculates the correct position of the description in the classification, inferring appropriate subsumption relationships.

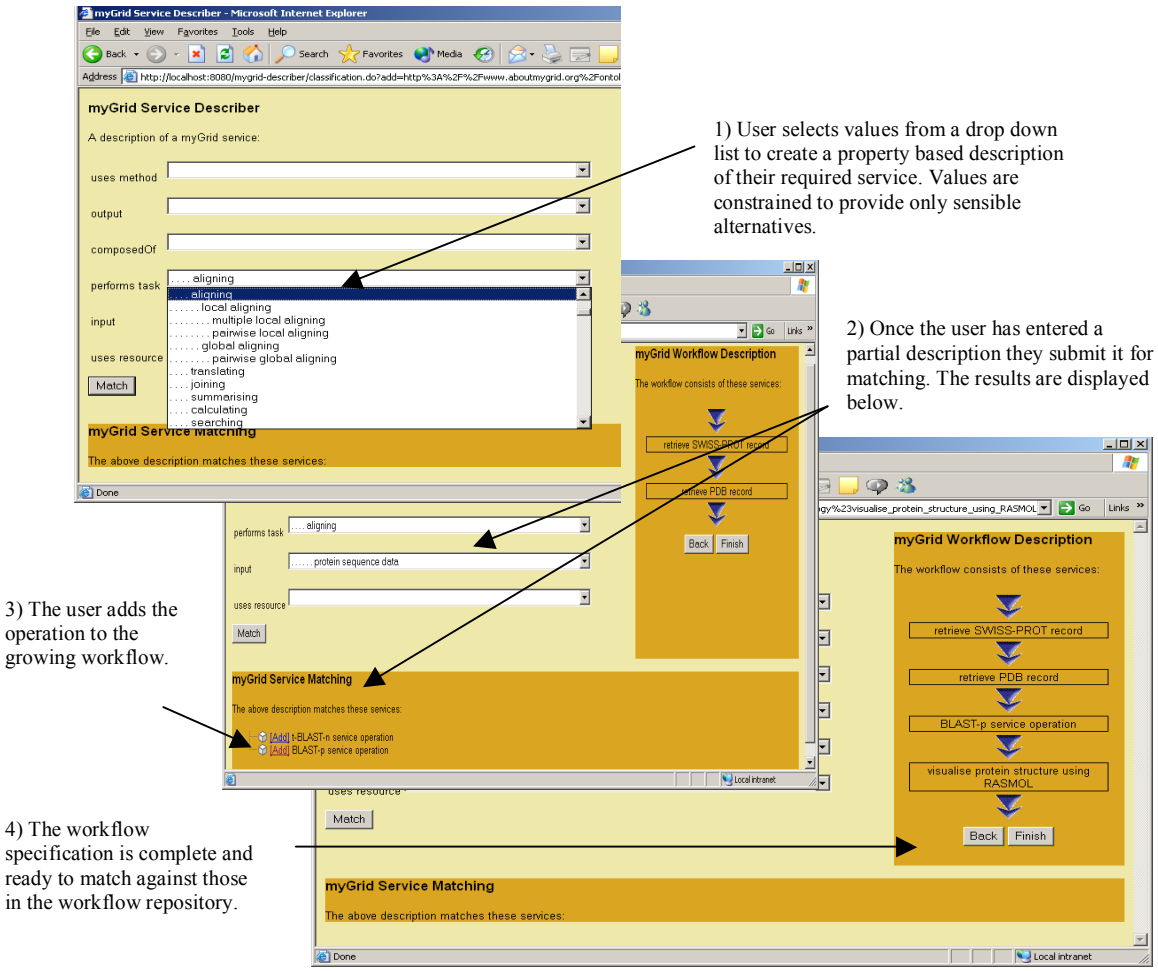

Figure 5. Screen shot of the myGrid portal showing the form with which a user can find services and construct simple workflows.

We envisage the user using a combination of partial description and hierarchy browsing to find the class of service they require. The user is therefore presented with the portion of the classification that fulfils their requirements, which they can browse further. Once they have found the specific class required, they can then select this to be the first step in a workflow. The current structure of the workflow is shown to one side.

In the same way that the semantic type of the initial data constrains the description of the initial step in the workflow, descriptions of subsequent steps have their input slots constrained to the output of the preceding step. For example, if the first step produces a SWISS-PROT record, the next stage must accept a SWISS-PROT record.

After an arbitrary number of steps the user will have specified a workflow which produces the required result, whether this is in the form of derived data to be stored in the their personal repository, or a graphical display of the results on their machine.

As the graphical representation of the framework is being constructed, an underlying representation in WSFL should also be constructed. This has yet to be implemented. At this stage the framework relies on the simple ontological description of the composed service that has in effect been created by the workflow. It compares this newly created composed service with composed services embodied in previously manually authored WSFL workflows. If a match is found based on subsumption testing, the results are presented to the user. Each workflow has associated with it metadata, both ontological and in other representations,

which aid the user in making a single choice. For example, one workflow may use static service binding (i.e. has already stated the end points for individual web services). Another may employ dynamic service binding using the service directory to resolve a class from the service classification to specific available service instances.

Once the user has selected the appropriate workflow, the WSFL specification together with the selected data is sent to the workflow enactment engine. The enactment engine uses the service classification to resolve an abstract specification of a service into a concrete service instance to invoke. If the step within a workflow is specified as a class of service rather than a specific service instance, the enactment engine consults the UDDI-M service directory. The directory searches for registered service instances, which are classified directly as the desired class. No tree walking or dynamic classification is undertaken, although this will be implemented in future prototypes. If more than one running service is available the enactment engine either arbitrarily takes the first or asks the user to choose. In future prototypes this choice will depend on information within individual service profiles.

The enactment engine stores results of the workflow in the user repository and annotates the results with their semantic type, ensuring the data is available for processing by further workflows.

4. Discussion

We have found through implementation in a substantial Grid based middle ware project myGrid, that semantic matching of bioinformatics web services can be accomplished successfully by subsumption reasoning based DAML+OIL service descriptions. Users successfully employed web based forms to capture their functional requirements for a web service and match against an existing service classification. The DAML+OIL infrastructure used to support the linking of description, classification and constraints was hidden from the user but the apparent behaviour was intuitive.

DAML-S proved to be a useful foundation for service descriptions. However, as in much of the current work on matching web services, it concentrates to a large extent on business and geographic aspects of a service description such as cost and location. Although important, the primary rationale for matching a bioinformatics service is based on functionality and we found it necessary to add properties to the DAML-S model to adequately capture these aspects of bioinformatics services for example the task performed.

We found DAML-S to be over inclusive particularly in regard to process models. DAML+OIL provides no intrinsic support for representing control structures or data flows. We found it far more productive to represent this information in the workflow language WSFL, for which we have access to an enactment engine. Recent papers by the DAML-S community also point to the need to translate process models into another representation for verification and enactment [23].

Linking classifications to descriptions requires an approach to ontology construction not commonly found in most current projects. Building an ontology becomes an exercise in description rather than manual classification. Although others have illustrated how the DAML+OIL technology can describe web services, implementing such an approach raises significant issues. For a pilot bioinformatics implementation we required significant sized description based ontologies just test the basic utility of the approach. The suite of ontologies included 580 concepts, over a half of which were fully defined using DAML+OIL expressions.

This descriptive effort is not confined to the initial developers of the ontology. Someone publishing their service in the myGrid framework must fully describe their service using DAML+OIL in order for it to be registered. This has major implications for the scope and coverage of the underlying ontologies and software needed to support the registration process, which apply not just to myGrid, but also to the Grid and the Semantic Web in general. Key points include:

Coverage sufficiency: finding the point when the ontologies contain sufficient content to adequately describe the majority of what bioinformaticians want to provide. Widely adopted ontologies within the biomedical area such as the Gene Ontology have tens of thousands concepts and are constantly growing.

Description articulation: how publishers and seekers of services write descriptions given that they will no experience of ontology authoring (nor should they need any). We hope to use a similar mechanism to the

web based forms for service matching. However, if the ontologies do not provide sufficient vocabulary to describe a service the distinction between service authoring and ontology authoring may become blurred. The publisher must be able to extend ontologies in limited ways but there are issues concerning how these extensions propagate through the system.

In myGrid.version0 we tended to describe services from the point of bioinformatics (e.g. SWISS-PROT) rather than more conceptually biology (e.g. alignment). Further work is needed to provide a full range of descriptions for the full range of users.

The workflows formed by the myGrid portal are services themselves, and as such have inputs, and outputs (which are easily inferable from the component services) and fuinctional descriptions generated from the component descriptions. These latter descriptions are cumbersome and are oriented to describing the process rather than the overall function of the workflow. Further work is needed on how to capture descriptions of workflows.

5. Acknowledgements

This work is supported by the myGrid eScience pilot (EPSRC GR/R67743), GONG (DARPA DAML subcontract PY-1149 from Stanford University) and the Geodise eScience pilot (EPSRC GR/R67705). The authors would like to acknowledge the rest of the myGrid team: Matthew Addis, Nedim Alpdemir, Rich Cawley, Vijay Dialani, David De Roure , Justin Ferris, Rob Gaizauskas, Kevin Glover, Chris Greenhalgh, Ian Horrocks, Peter Li, Xiaojian Liu, Phillip Lord, Darren Marvin, Simon Miles, Luc Moreau, Tom Oinn , Norman Paton, Stephen Pettifer, Milena Radenkovic, Alan Robinson, Tom Rodden, Martin Senger, Nick Sharman, Brian Warboys, Paul Watson.

6. References

1 Goble C, Stevens R, Ng G, Bechhofer S, Paton N, Baker P, Peim M and Brass A. *Transparent access to multiple bioinformatics information sources.* IBM Systems Journal, Vol. 40, No. 2, pp 532-551, 2001.

2 Foster I, Kesselman C, and Tuecke S. *The anatomy of the Grid: Enabling scalable virtual organizations*. In The International Journal of Supercomputer Applications, 2001.

3 Hass L, Schwarz PM, Kodali P, Kotlar E, Rice JE and Swope WC *DiscoveryLink: A system for integrated access to life sciences data sources.* IBM Systems Journal, Vol. 40, No. 2, pp 489-511, 2001.

4 Altschul, SF, Gish W, Miller W, Myers EW & Lipman DJ (1990) *Basic local alignment search tool.* J. Mol. Biol. 215:403-410.

5 Horrocks I. DAML+OIL: a reason-able web ontology language. In *Proc. of EDBT 2002*, March 2002.

6 Ankolekar A, Burstein M, Hobbs J, Lassila O, Martin D, McIlraith S, Narayanan S, Paolucci M, Payne T, Sycara K, Zeng H) *DAML-S: Semantic Markup for Web Services* in Proceedings of the International Semantic Web Working Symposium (SWWS), July 30-August 1, 2001.

7 Rice P, Longde I, and Bleasby A. *EMBOSS: The European Molecular Biology Open Software Suite* Trends in Genetics June 2000, vol 16, No 6. pp.276-277

8 http://www.rosettanet.org/

9 http://www.unspsc.org/

10 http://www/uddi.org/

11 Siepel AC, Tolopko AN, Farmer AD, Steadman PA, Schilkey FD, Perry BD, Beavis WD An integration platform for heterogenous bioinformatics software components in IBM Systems Journal, Vol. 40, No. 2, pp 570-591, 2001.

12 Baru C, Moore R, Rajasekar A, Wan M The SDSC Storage Resource Broker, *Proc. CASCON'98 Conference*, Nov 30-Dec 3, 1998, Toronto, Canada

Conference , Nov.30-Dec.3, 1998, Toronto, Canada

13 BioMOBY http://www.biomoby.org/. 2002

14 Paolucci M, Kawamura T, Payne TR, and Sycara K, *Semantic Matching of Web Services Capabilities*, to appear in The First International Semantic Web Conference (ISWC), June, 2002.

15 Foster I, Kesselman C, Nick J, Tuecke S The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.; January, 2002. from http://www.globus.org/ogsa/

16 The International Statistical Classification of Diseases and Related Health Problems, tenth revision http://www.who.int/whosis/icd10/

17 http://www.geneontology.org/

18 http://www.snomed.org/

19 Berners-Lee T, Hendler J and Lassila O The Semantic Web in Scientific American pp: 28-37, May 2001.

20 Horrocks I. The FaCT system. In H. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98* in Lecture Notes in Artificial Intelligence 1397, pages 307-312. Springer-Verlag, Berlin, May 1998.

21 Universal description discovery and integration (UDDI). http://www.uddi.org, 2001.

22 Stevens R, Horrocks I, Goble C, Bechhofer S *Building a Reason-able Bioinformatics Ontology Using OIL*. IJCAI'01 Workshop on Ontologies and Information Sharing, Seattle, USA, pp. 81--90, August 2001.

23 Narayanan S, McIlraith S.A. *Simulation, Verification and Automated Composition of Web Services.* The Eleventh International World Wide Web Conference WWW2002. pp 77-88 (2002)

24 Trastour D, Bartolini C, Preist C *Semantic Web Support for the Business-to-Business E-Commerce Lifecycle* The Eleventh International World Wide Web Conference WWW2002. pp: 89-98 (2002)

25 Bechhofer S, Horrocks H, Goble C, Stevens R. *OilEd: a Reason-able Ontology Editor for the Semantic Web*. Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September 19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396--408. 2001

26 WSFL 1.0 http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

27 XLANG http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

28 Carr L, Bechhofer S, Goble C, Hall W *Conceptual Linking: Ontology-based Open Hypermedia.* WWW10, Tenth World Wide Web Conference, Hong Kong, May 2001

29 Rector AL, *Clinical terminology: Why is it so hard?*, Methods of Information in Medicine, pp. 239-252, ISSN 0026-1270, Vol. 38, December, Schattauer.

30 Ashburner M et al *Gene Ontology: tool for the unification of biology*. Nature Genetics 25: 25-29 (2000)

31 Bechhofer S, Horrocks I. *Driving User Interfaces from FaCT*. International Workshop on Description Logics (DL 2000), RWTH Aachen, Germany August 17--19, 2000.

32 Rogers JE and Rector AL (1996). *The GALEN ontology*. Medical Informatics Europe (MIE 96), Copenhagen, IOS Press: 174-178.

33 Bechhofer S, Goble C, *Classification Based Navigation and Retrieval for Picture Archives*. IFIP WG2.6 Conference on Data Semantics, DS8 , Rotorua, New Zealand, Jan 1999

34 Baker PG, Goble C, Bechhofer S, Paton N, Stevens R, Brass A. *An Ontology for Bioinformatics Applications*. Bioinformatics, *15*(6) pp 510--520, 1999.