Semantic Integration in ACGT

Project Number:	FP6-2005-IST-026996
Deliverable id:	D 9.4
Deliverable name:	Semantic Integration in ACGT
Date:	April 6, 2009

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	D 9.4
Document name:	Semantic Integration in ACGT
Document type (PU, INT, RE)	INT
Version:	1.0
Date:	6/4/2009
Editors: Organisation: Address:	Stelios Sfakianakis FORTH-ICS Foundation for Research and Technology-Hellas (FORTH) Institute of Computer Science N. Plastira 100, Vassilika Vouton, GR-700 13 Heraklion, Crete, Greece

Document type PU = public, INT = internal, RE = restricted

ABSTRACT:

This document aims to describe the ACGT infrastructure aiming at achieving semantic interoperability among the services and tools. We present existing standards and investigate their applicability in the context of the ACGT platform. Emphasis is particularly given on what appear to be the practical needs of the ACGT users rather than to generic but more costly solutions. For that reason we present the BioMOBY services and relevant ontologies both as the main source providing semantic information about domain specific data types and as a case study for the semantics enabled service discovery and workflow construction. We finally define a generic semantic service integration architectural framework using standard Semantic Web technologies, standards, and tools. Such a framework provides the semantic integration of not only ACGT services but also other third party services like BioMOBY.

KEYWORD LIST: semantics; web services; semantic interoperability; integration

MODIFICATION CONTROL			
Version	Date	Status	Author
0.1	15/03/2009	Draft	Stelios Sfakianakis
0.2	2/4/2009	Draft	Stelios Sfakianakis
0.5	6/4/2009	Pre-Final	Stelios Sfakianakis
1.0	6/4/2009	Final	Stelios Sfakianakis

List of Contributors

- Stelios Sfakianakis, FORTH-ICS
- Lefteris Koumakis, FORTH-ICS
- Manolis Tsiknakis, FORTH-ICS
- Stefan Rueping, FhG
- Thierry Sengstag, SIB
- Johan Karlsson, UMA
- Oswaldo Trelles, UMA

Contents

Table of Figures

Acronyms

API	Application Programming Interface
BPEL	Business Process Executable Language
CA	Certification Authority
DMS	Data Management Service
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
N3	Notation 3 serialization format for RDF
OWL	Web Ontology Language
OWL-S	OWL for Services
OWL-DL	OWL Description Language
RDF	Resource Description Framework
RDFS	RDF Schema
SAWSDL	Semantic Annotations for WSDL
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
WS	Web Services
WSDL	Web Service Description Language
WSMO	Web Services Modeling Ontology
WSMF	Web Services Modeling Framework
XHTML	Extensible Hyper Text Markup Language
XML	Extensible Markup Language

Executive Summary

The ACGT environment provides a unified architecture where data, processing and visualization tools, and knowledge discovery services cooperate in order to fulfil the end user goals and scenarios. The Workflow Environment in particular is a central suite of components where such cooperation is tested and validated every time the users try to build novel scientific workflows and relevant experiments. The concept of service interoperability and especially the kind of it that relates to semantics is therefore of utmost importance so that the integration of the different ACGT software actors that participate in the complex user workflows is feasible. In this deliverable we focus on the semantics integration of the ACGT services, which aims to ease the construction of meaningful workflows, i.e. workflows that not only execute successfully but also function and produce results that make sense according to the underlying domain knowledge of the users.

To achieve this goal, Semantic Web technologies are the prominent tools, given not only their current endorsement and support by standardization bodies like W3C but also their sound theoretical foundations and their inherent compliance with the universal Web infrastructure. This document therefore provides details for the design of a Semantic Web compliant infrastructure that caters for the ACGT specific semantic annotation, discovery, and orchestration of services and tools. To a large extent previous results and deliverables such as the Metadata Repository and the Workflow Environment are part of this infrastructure and for that reason the current document aims to provide a consolidated view of the ACGT service oriented architecture, with special focus on the semantics, and to fill in any missing pieces.

A particular objective of this work is to offer links to existing standards and to investigate their applicability in the context of the ACGT platform. Nevertheless, the abundance of the available information in the field in terms of the specifications and the relevant efforts led us to consider a more limited set of them that appear to be “state of the art” and more close to standardization. An additional important factor is the emphasis on what appear to be the practical needs of the ACGT users rather than to generic but more costly solutions. For that reason we present the BioMOBY services and relevant ontologies both as the main source providing semantic information about domain specific data types and as a case study for the semantics enabled service discovery and workflow construction.

The main outcome of this work is the definition of a generic semantic service integration architectural framework using standard Semantic Web technologies, standards, and tools. Such a framework provides the semantic integration of not only ACGT services but also other third party services like BioMOBY.

1 Introduction

The ACGT platform aims to facilitate the seamless and secure access and analysis of multi-level clinico-genomic data using high-performing knowledge discovery operations and services. In order to achieve this goal, a well defined data analysis and processing environment needs to be in place, which would make possible the integration and interoperability of the different ACGT components. The goal of the integration process is to make disparate and heterogeneous applications work together so as to produce a unified set of functionality, possibly by complementing each other. Whereas integration is concerned with the building of a unified system that incorporates the functionality of its constituent parts, interoperability is more a virtue of a single software entity so that it can be easily deployed in an unanticipated environment. Therefore defining interoperability guidelines is a prerequisite for building the ACGT integrated environment.

In ACGT two notions of interoperability have been specified (see Deliverable 9.1): the syntactic and semantic interoperability. Syntactic interoperability of software may be defined as the ability for multiple software components to interact regardless of their implementation programming language or hardware platform. Syntactic interoperability in ACGT requires standardization of data formats and data structures for the representation of, access to and exchange between biomedical informatics resources. On the other hand, semantic interoperability is related to the "meaning" of the exchanged information and it is the ability of two or more interacting computer systems to have the meaning of that information accurately and automatically interpreted and "understood". To achieve syntactic interoperability programming and messaging interfaces must conform to standards that specify consistent syntax and format across all systems in the ACGT environment. Furthermore, in order to support the semantic interoperability, all data must be annotated with metadata by means of terminology and ontology identifiers and codes that support aggregation, comparison, summarization, mining, etc. of information that resides in separate resources.

The complexity and the diversity of user requirements have a strong impact on the design of the ACGT architecture. This architecture has been early defined as "service oriented". In a service oriented environment the service is the central entity. When we use the term service in this context we mean a software component that is capable of performing certain tasks for other services or the principal human user. A web service in particular is a software entity that is accessed through the ubiquitous web infrastructure and its related protocols and machine readable formats like XML. The adopted architecture for ACGT is therefore built

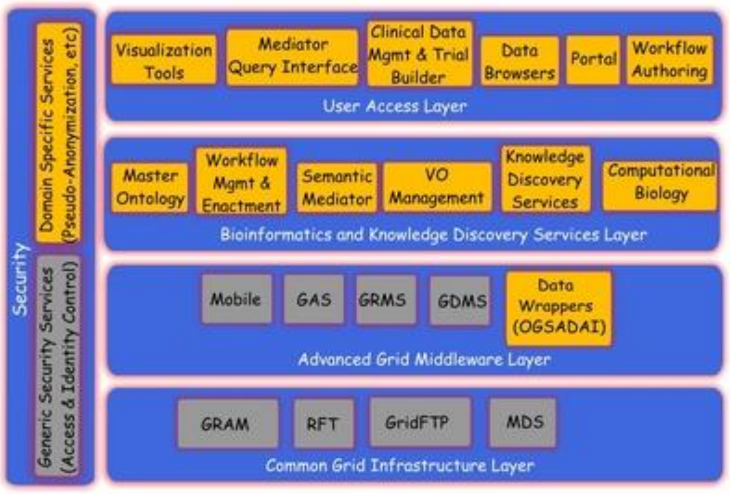

Security
Domain Specific Services (Pseudo-Anonymization, etc)
Generic Security Services (Access & Identity Control)
Visualization Tools
Mediator Query Interface
Clinical Data Mgmt & Trial Builder
Data Browsers
Portal
Workflow Authoring
User Access Layer
Master Ontology
Workflow Mgmt & Enactment
Semantic Mediator
VO Management
Knowledge Discovery Services
Computational Biology
Bioinformatics and Knowledge Discovery Services Layer
Mobile
GAS
GRMS
GDMS
Data Wrappers (OGSADAI)
Advanced Grid Middleware Layer
GRAM
RFT
GridFTP
MDS
Common Grid Infrastructure Layer

Figure 1 The ACGT Architecture

around the notion of services and a view of it is shown in Figure 1. A layered approach has been followed for providing different levels of abstraction and a classification of functionality into groups of homologous software entities. In this approach we consider the security services and components to be pervasive throughout ACGT so as to provide both for the user management, access rights management and enforcement, and trust bindings that are facilitated by the grid and domain specific security requirements like pseudonymization. Apart from the security requirements, the grid infrastructure and other services are located in the first (lowest) two layers: the Common Grid Layer and the Advanced Grid Middleware Layer. The upper layer is where the user access services, such as the portal and the visualization tools, reside. Finally, the Bioinformatics and Knowledge Discovery Services are the "workhorse" of ACGT and the corresponding layer is where the majority of ACGT specific services lie.

For the realization of this architecture a multidisciplinary and multi paradigm approach has been followed. The ACGT platform is designed according to the following technologies and standards: Service Oriented Architecture (Web Services), the Grid, and the Semantic Web. In particular, Grid and Web Services technologies are the basis for defining the syntax and structure of the exchanged messages to achieve syntactic interoperability:

- The machine to machine communication is performed via XML programmatic interfaces over web transport protocols (SOAP), which are specified using the Web Service Definition Language (WSDL). These common data representation and service specification formats, when properly deployed, make the syntactic integration of the ACGT components a lot easier.
- The Grid defines the general security framework, the virtual organization abstraction, the user management mechanisms, authorization definition and enforcement etc. It also provides the computational and data storage infrastructure that is required for the management and processing of large clinical and genomic data sets.

On the other hand, the Semantic Web provides the infrastructure for the semantic interoperability: it adds the knowledge representation mechanisms by the means of RDF Schemas and OWL ontologies, the unique identification of concepts and resources through the URIs, the implementation-neutral query facilities with the SPARQL "universal" query language and the associated query interfaces, etc. It is the aim of this deliverable to support the use of such technologies in the ACGT platform for providing more user friendly, intelligent and advanced system behaviour.

The need for semantics and the deployment of Semantic Web compliant technologies are guided by specific user requirements. It is usual the case that, when using services, scientists need to:

- Find them, i.e. locate them irrespective of their location or providing (hosting) organization
- Interpret them – what do the services do or provide, what kind of "experiments" the end users can perform by using them
- Know how to invoke them – what data and initial parameters do they need to supply
- Know their behaviour – how they should be invoked (choreography) or being composed (orchestration) to achieve a higher level goal or scenario.

To this end semantic web services enable automation and greatly facilitate:

- Service Selection and Invocation
- Translation of messages and Mediation between different services
- Service Composition

- Monitoring and Recovery
- Contracting, Negotiation, verification, simulation, etc.

In this document we have selected the service *discovery*, *selection*, and "*matchmaking*" as the primary use cases where semantics descriptions for services fit in. All of these are advanced features of a modern problem solving environment such as the Workflow Editor and Enactment environment that the ACGT, and in particular work package 9, aims to deliver.

The rest of the document gives some background information about the Semantic Web stack of technologies in Section 2, and then proceeds to survey some of the proposed standards and technologies for the semantic annotation and discovery of services. The last part of the document presents the definition of a generic semantic service integration architectural framework using standard Semantic Web technologies, standards, and tools. Such a framework provides the semantic integration of not only ACGT services but also other third party services like BioMOBY.

2 An "express course" in the Semantic Web

The Semantic Web (Berners-Lee et al., 2001) aims to support the representation and exchange of information in a meaningful way so as to make possible the automated processing of descriptions on the Web. The objective is to enrich the unstructured information in the current Web with machine processable descriptions of the semantics in order to make its navigation and exploration by software agents as easy as it's for the human users today, or even easier. In this context Semantic Web promotes a shift from the current "syntactic" world to the future "semantic" world of services, applications, and people and aims to make the machine to machine communication feasible so that not only data but also information and finally knowledge are shared.

In technological terms the Semantic Web architecture consists of an array of technologies that can roughly be visualized in a layered design layout as depicted in Figure 2. The basic infrastructure in the bottom layers in this stack of technologies is the exactly the same to the syntactic web: Uniform Resource Identifiers (URIs) used for identification of web resources, universal encoding schemes for characters, i.e. Unicode, and XML and its related technologies (e.g. XML Namespaces) as a ubiquitous data serialization format. Some of the upper layers like Proof and Trust are missing or are work in progress. Here we will concentrate on the middle layers where the core infrastructure technologies of the Semantic Web reside: RDF, RDF Schema/OWL, and SPARQL.

The Resource Description Framework (RDF) is a syntax neutral data model that enables the description of web resources in a simple way (Lassila, Swick, et al., 1999). At the core of RDF there is a model for representing and describing resources through named properties (also known as predicates) and their values. The resources can be anything that can be identified with a URI. Although in the initial specification of RDF resources were limited to web documents and web sites, it is possible and quite frequent in practice to describe, by the means of RDF and the various URI schemes, real world entities like people, or more abstract things like relationships and concepts. The use of use of URIs and especially the HTTP based ones for identifying persons or other physical entities may seem strange at first but this is in compliance with the architecture of the World Wide Web (Berners-Lee et al., n.d.) which strongly suggests the use of URIs for identifying anything that can be of importance irrespective of how abstract or tangible it may be.

The properties serve both to represent attributes of resources and to represent relationships between resources. They are also identified though URIs to make them unique. The combi-

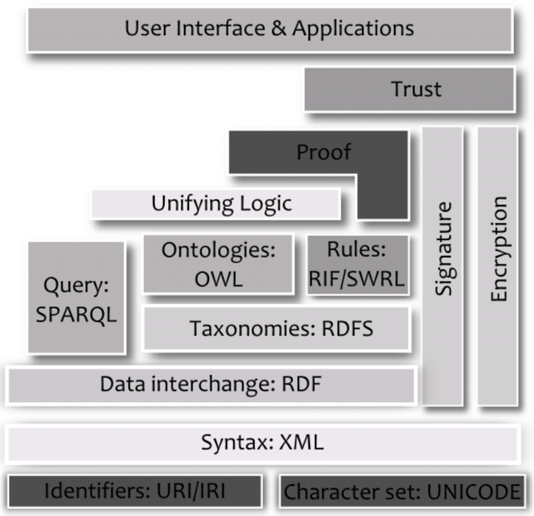


Figure 2 The Semantic Web stack of technologies

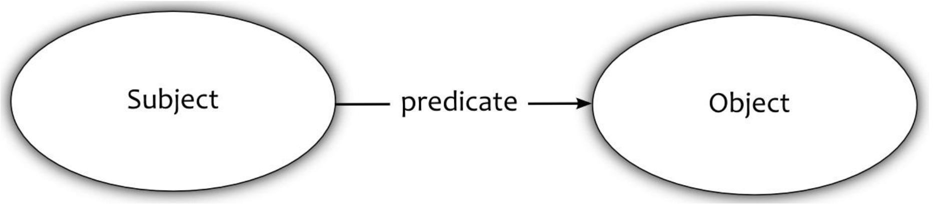


Figure 3 RDF Data Model

nation of resources and the properties that connect them builds the simple RDF data model. In this data model the primary informational building block is the "triple" which denotes the subject – property - object expressions (Figure 3). The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. Since an object of a triple can be the subject of another one, a set of RDF triples forms a directed graph where the RDF resources, both subjects and objects, are the nodes of the graph and the predicates are the labelled arcs.

RDF as an abstract model is independent of any specific serialization syntax. The normative representation syntax for RDF graphs is XML but more lightweight formats, such as Turtle (Beckett & Berners-Lee, 2008), exist. Using such a simpler format we can give information about a person in the following way[1]:

```
me:stelios foaf:givenname "stelios" .
me:stelios foaf:family_name "Sfakianakis" .
me:stelios foaf:workplaceHomepage <http://www.ics.forth.gr/cmi-hta> .
```

Or in a more condensed format (where we have grouped the RDF triples referring to the same subject):

```
me:stelios foaf:givenname "stelios"
                  ; foaf:family_name "Sfakianakis"
                  ;  foaf:workplaceHomepage <http://www.ics.forth.gr/cmi-hta> .
```

The simplicity and flexibility of RDF is evident but in certain cases its generality must be formally confined so that software entities are able to correctly exchange the encoded information. For example, stating that an animal is the creator of a web page does not make sense in the real world but RDF does not forbid anyone for making such a claim. Ontologies (Uschold & Gruninger, 1996) provide such a tool to specify what can be expressed in the context of an application domain or in a real world scenario, what is the underlying meaning, and how the information presented can be further processed to generate more information. Moreover ontologies and their less powerful relatives like taxonomies and thesaurus provide the means for achieving a common interpretation of a domain and a shared understanding of the concepts and relationships involved. In the Semantic Web there are two main technologies for providing such rigor: RDF Schema and OWL (Brickley & Guha, 2004; Dean, Schreiber, et al., 2004). RDF Schema provides the means for defining classes, class hierarchies, properties, property hierarchies, and property restrictions. Its expressive power is basically limited to the representation of concepts, their relations, and taxonomies of concepts. On the other hand the Web Ontology Language (OWL) was introduced to address the need for more expressiveness and extends the RDF Schema by providing three variants: OWL-Lite, OWL-DL, and OWL-Full. Without delving into details, the different species of OWL provide different degrees of expressiveness and are able to define existential restrictions, cardinality constraints in properties, property types like inverse, transitive, and symmetric, and a lot more.

[1] me: and foaf: are namespace bindings (their definitions are not shown in these examples) to provide abbreviated URI references.

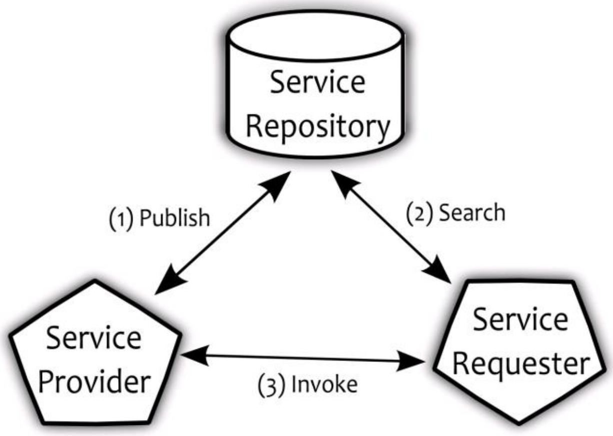


Figure 4 Web Service interaction "protocol"

The added features of OWL allow the ontologies built in conformance to it to be formally treated and the data represented are amenable to "reasoning" and inference, i.e. they can be processed according to formal logic rules to deduce new information. All these happen on the basis of the web infrastructure: RDF resources and their URI references are used, the open world assumption is followed, since partial information on the Web is a quite frequent phenomenon, and the ontologies themselves can be freely intermixed and meshed since hyperlinks are employed everywhere.

Since RDF is the common interchange and representation model of information, the Semantic Web transforms the hyperlinked syntactic World Wide Web to a huge database or a Global Giant Graph, as Tim Berners-Lee put it. The standard query language for this huge database is SPARQL (Prudhommeaux & Seaborne, 2008), which is similar to SQL. In addition to the query language the SPARQL standard defines an application protocol for the submission of queries to RDF sources and the retrieval of results. With the query language and the access protocol defined, the SPARQL specifies a web friendly interface to RDF information, whether this is actually stored as RDF triples or not. It is therefore feasible to make SPARQL queries to relational or other databases through an appropriate wrapper or transformation process that translates, either online or in some pre-processing step, the internal data to an RDF compliant format. As a result these Semantic Web technologies enable the connection of data between different and heterogeneous data sources, effectively allowing data in one data source to be linked to data in another data source (Bizer, Heath, Idehen, & Berners-Lee, 2008).

3 Semantic Web Services: Technologies and Standards

In the prototypical Web Service use case scenario shown in Figure 4 a "Service Requester" locates the available services by searching in a "Service Repository" (or Registry) where the services have been advertised by storing there their descriptions. This is actually the exact scenario followed in the ACGT platform as well, where the Metadata Repository is the central service registry.

In order for such scenarios to take place, services should be annotated and described in the most appropriate way so that they are easily discovered and used. As shown in Figure 5, we can again identify two levels in the services descriptions: the semantic and the non-semantic level. The Semantic Service Stack adopts the following general types of service contracts (Sheth, 2003):

- Information Model defines the data model for the input, output and fault messages of the services.

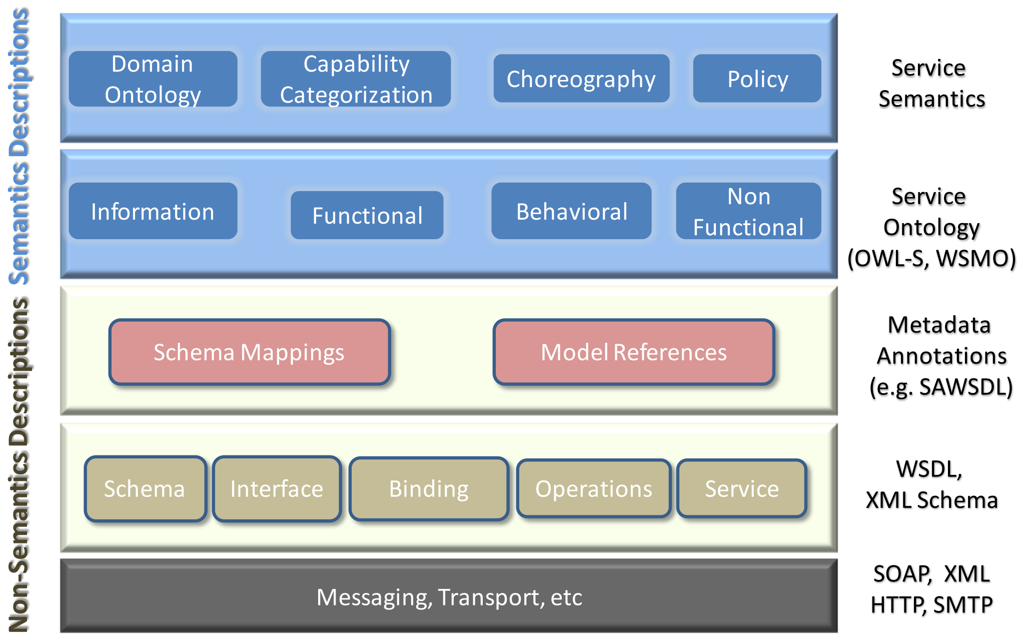


Figure 5 “Stack" of technologies for Semantic Web Services

- Functional Descriptions define service functionality and its capabilities, i.e. what a service provide to its callers.
- Non-Functional Descriptions define additional aspects of the service implementation and environment such as “quality of service” (performance, throughput, accuracy, etc) or policies, e.g. security.
- Behavioural Descriptions define the external and internal behaviour of the service. The externally visible behaviour (“choreography”) is for example the “protocol” the client has to follow when contacting the service, e.g. the sequence of operation invocations. On the other hand the internal behaviour is related to the way the service is implemented by the composition and orchestration of other services.
- Technical Descriptions define messaging details, such as message serializations, communication protocols, and physical service access points.

There are a number of technologies, specifications, and efforts to define parts of the above aspects or to cover all of them in one unified approach. In the following subsections we survey some of them that have gathered some momentum over the last couple of years. We also propose the adoption of one of them and its customization in the context of ACGT.

3.1 WSMO: Web Services Modelling Ontology

Web Service Modelling Ontology (WSMO) is an ontology for semantically describing Semantic Web Services (Roman et. al, 2005). It is a model for the description of semantic web services that tries to overcome the limit of the existing technologies for the service description. Web Service Modelling Language (WSML) is a language that formalizes the WSMO. It uses well-known logical formalisms, namely, Description Logics, First-Order Logic and Logic Programming, in order to enable the description of various aspects related to Semantic Web Services. It consists of a number of language variants with different underlying logic formalisms.

The conceptual grounding of WSMO is based on the Web Service Modelling Framework (WSMF, see Figure 6), wherein four main components are defined.

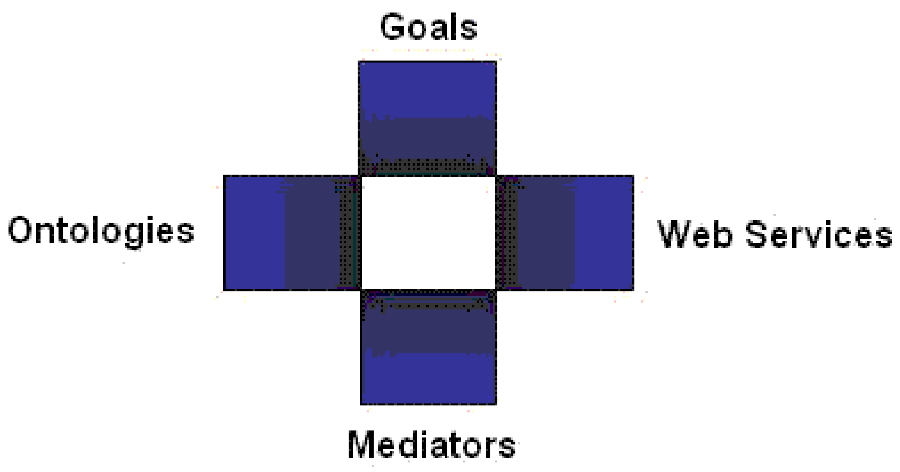


Figure 6 Web Services Modelling Ontology

- Ontologies provide machine-readable semantics for the information used by all actors implied in the process of Web Services usage, either providers or consumers, allowing interoperability and information interchange among components.
- Goals specify objectives that a client may have when consulting a Web Service. A goal in WSMO is described by non-functional properties, postconditions, and effects. Non-functional properties specify information that do not affect the functionality of the element, including for example quality-related attributes. Post-conditions define the state of the desired information space. Effects describe the desired state of the world after the execution of the Web Service.
- Web Services represent the functional part that must be semantically described in order to allow its (semi-)automated use. In a WSMO specification, Web Services are described by means of non-functional properties, imported ontologies, used mediators, capability and interfaces. A service can be described by multiple interfaces, but has one and only one capability.
- Mediators aim to overcome structural, semantic or conceptual mismatches that appear between different components that build the WSMO specification. Mediators are used as connectors to provide interoperability facilities among the rest of components. Mediation within Semantic Web Services can be done at different levels: data level is mediation between heterogeneous data sources; protocol level is mediation between heterogeneous communication patterns; process level is mediation between heterogeneous business processes.

WSMO is also working on the definition of a set of use cases in order to exemplify WSMO usage for specific real-life purposes. The different use cases provide valuable insight for testing and adapting the modelling constructs provided in WSMO in real-world scenarios for Web Services. So, besides demonstrating how to model Web Services in WSMO, the use cases also allow demonstration of the adequacy of the WSMO approach in terms of providing an exhaustive framework for covering all relevant aspects of semantic description of Web Services.

3.2 OWL-S: OWL for Services

OWL-S[2] (formerly DAML-S) builds on top of OWL and allows for the description of a Web service in terms of a Profile, which tells "what the service does/provides", a Process Model, which tells "how the service works", and a Grounding, which tells "how to access the service" (Martin et. al, 2004). The service profile describes what is accomplished by the service, any limitations on service applicability and quality of service, and requirements that the service requester must satisfy in order to use the service successfully. The process model gives details about the semantic content of requests, the conditions under which particular outcomes will occur, and, where necessary, the step by step processes leading to those outcomes. In the process model a service can be described as an atomic process that can be executed in a single step or a composite process that, similar to a workflow, can be decomposed in other processes based on control structures like 'if-then-else' and 'repeat-while'. Finally, Grounding descriptions supply information about the communication protocol and other transport information (such as port numbers) and the message formats and serialization methods used in contacting the service.

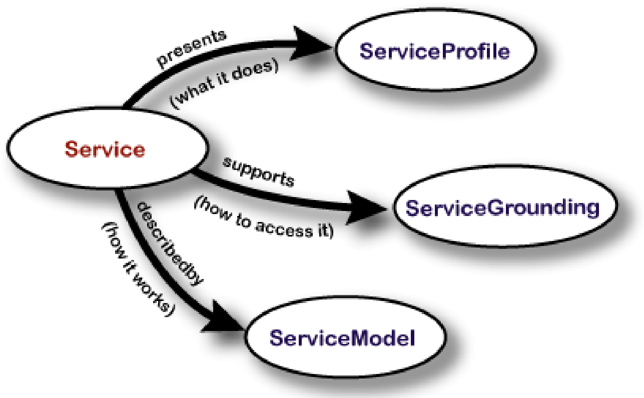


Figure 7 The top level of the OWL-S ontology

[2] http://www.daml.org/services/owl-s/

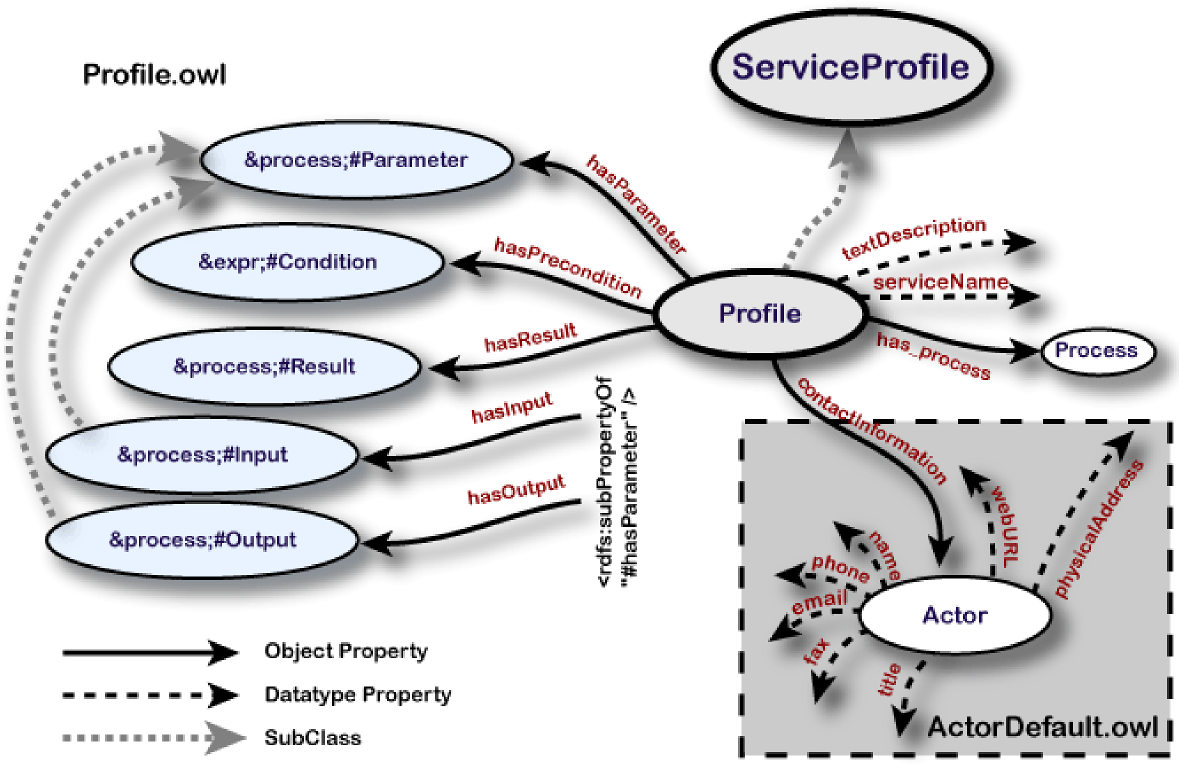


Figure 8 The OWL-S Service Profile ontology

The OWL-S profile provides a set of concepts to specify capabilities of services, with the goal of supporting capability-based discovery. Specifically, the OWL-S profile allows service providers to advertise what their services do, and service requesters to specify what capabilities they expect from the services they need to use (see Figure 8). Crucially, the profile provides an explicit description of those capabilities, so that they do not have to be extracted from incidental properties such as the name of the service, or the company that provides it. By exploiting the structure of OWL-S profiles and their references to OWL concepts, a discovery process can find those services that are most likely to satisfy the needs of a requester.

3.3 Semantic Annotations for WSDL

In 2006, the W3C created a charter for the Semantic Annotation of Web Services (SAWSDL), which used WSDL-S as its primary input (Kopecký et. al, 2007). SAWSDL became a W3C candidate recommendation in January 2007.

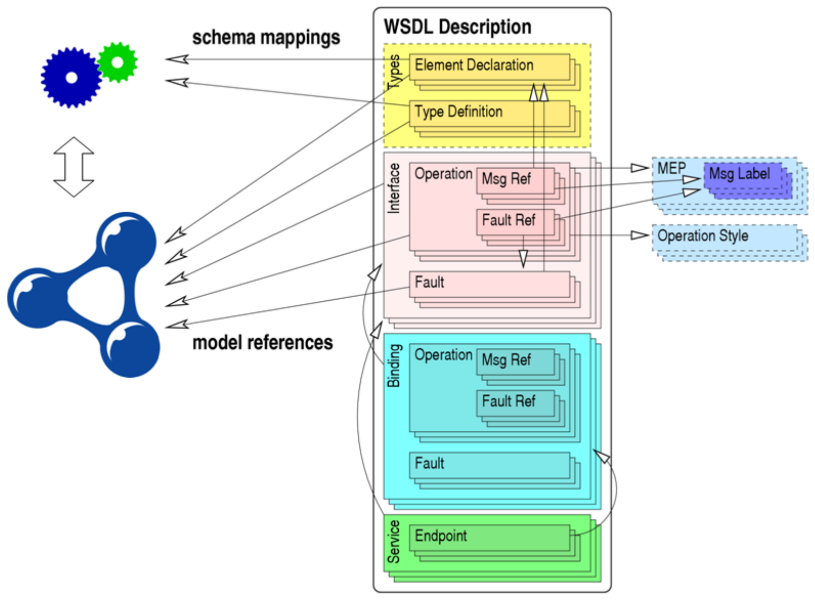


Figure 9 The Semantic Annotations for WSDL

SAWSDL defines mechanisms using which semantic annotations can be added to WSDL components. SAWSDL defines how to add semantic annotations to various parts of a WSDL document such as input and output message structures, interfaces and operations. The extension attributes defined in this specification fit within the WSDL 2.0 extensibility framework. It provides mechanisms by which concepts from the semantic models that are defined either within or outside the WSDL document can be referenced from within WSDL components as annotations. These semantics when expressed in formal languages can help disambiguate the description of Web services during automatic discovery and composition of the Web services. For example, the specification defines a way to annotate WSDL interfaces and operations with categorization information that can be used to publish a Web service in a registry. The annotations on schema types can be used during Web service discovery and composition. In addition, SAWSDL defines an annotation mechanism for specifying the data mapping of XML Schema types to and from an ontology; such mappings could be used during invocation, particularly when mediation is required. To accomplish semantic annotation, SAWSDL defines extension attributes that can be applied both to WSDL elements and to XML Schema elements.

Semantic annotations are references from an element within a WSDL or XML Schema document to a concept in an ontology or to a mapping. The specification defines annotation mechanisms for relating the constituent structures of WSDL input and output messages to concepts defined in an outside ontology. Similarly, it defines how to annotate WSDL operations and interfaces. Further, it defines an annotation mechanism for specifying the structural mapping of XML Schema types to and from an ontology by means of a reference to a mapping definition. The annotation mechanism is independent of the ontology expression language and this specification requires no particular ontology language. It is also independent of mapping languages and does not restrict the possible choices of such languages.

The key design principles for SAWSDL are:

- The specification enables semantic annotations for Web services using and building on the existing extensibility framework of WSDL.
- It is agnostic to semantic representation languages.

- It enables semantic annotations for Web services not only for discovering Web services but also for invoking them.

Based on these design principles, SAWSDL defines the following three new extensibility attributes to WSDL elements to enable semantic annotation of WSDL components:

- an extension attribute, named modelReference, to specify the association between a WSDL component and a concept in some semantic model. This modelReference attribute can be used especially to annotate XML Schema type definitions, element declarations, and attribute declarations as well as WSDL interfaces, operations, and faults.
- two extension attributes, named liftingSchemaMapping and loweringSchemaMapping, that are added to XML Schema element declarations and type definitions for specifying mappings between semantic data and XML.

These mappings can be used during service invocation. Fortunately, SAWSDL is agnostic to both the domain model, which gives it a lot of flexibility: domain models can be as simple as agreed upon English-language terms or as complex as expressive ontologies that use formal models such as description logics.

3.4 BioMOBY Object and Service Ontologies

BioMOBY[3] is a Web Service interoperability initiative in the field of bioinformatics aiming to facilitate the integration of web-based bioinformatics resources. Currently there are two approaches to achieve such integration: The first approach, based on the Web Services paradigm, is referred to as "MOBY Services" (MOBY-S), while the second one is called "Semantic MOBY" (S-MOBY[4]) and is based on concepts from the Semantic Web. MOBY-S uses a set of simple, end-user-extensible ontologies as its framework to describe data semantics, data structure, and classes of bioinformatics services. These ontologies are shared through a Web Service registry system, MOBY Central, which uses the ontologies to semantically bind incoming service requests to service providers capable of executing them. S-MOBY on the other hand employs RDF and OWL and the document oriented infrastructure of the WWW (the GET/POST methods of HTTP) for publishing and retrieving information from its discovery servers.

The key difference between classic web services and BioMOBY services is in the definitions of the input/output data structures. Web services utilize XML schema to describe the basic syntax. For example an interface of a web service might define a String as one input parameter. However there is no way for another service or program to determine if that String is intended to be a DNA Sequence or any of the data-types that are commonly represented as strings. To overcome this problem, BioMOBY defines a user extensible ontology of bioinformatics data-types (Wilkinson M.D. et. al, 2008). Syntactic types are defined by a GO-like ontology, a simple structure with node connected by edges (Figure 10). Each node is a data class name and each edge defines the relationships between two classes.

The Object Ontology currently consists of over 300 different data syntax definitions, including many of the common legacy flat-file formats, as well as novel objects that have been constructed de novo by participating service providers. All these objects are described the same way: In contrast to generic data models like RDF, the BioMOBY Object Ontology limits the relationship types to the following ones:

[3] http://www.BioMOBY.org/

[4] http://www.semanticmoby.org/

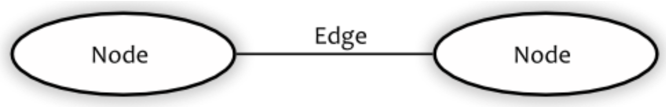


Figure 10 Moby Object Ontology

- "IS-A" is an inheritance relationship indicating that the first node (i.e. the "child") of the triple is a sub-class of the second node (i.e. the "parent"). All relationships of the "parent" are inherited and have the same semantics in the "child".
- "HAS-A" is a container relationship of "exactly 1" and indicates that the subject of the triple is defined as containing exactly one instance of the designated object. This is a transitive property.
- "HAS" is a container relationship with "1 or more" and indicates that the subject of the triple is defined as containing at least one instance of the designated object. This is a transitive property as well.

Figure 11 shows a sample of BioMOBY Object Ontology. All MOBY Objects inherit from the root "Object" Class, and since complex objects can only be derived through inheritance from (IS-A), or combination of (HAS-A/HAS) existing objects, every sub-object in a complex object is, itself, a valid MOBY Object which inherits directly or indirectly from the "Object" Class. Since all sub-components of all data-types are themselves BioMOBY Objects, generic reusable software is capable of extracting and/or assembling the data components of any possible BioMOBY object, including objects that did not exist when that software was created.

The root class of the ontology – "Object" – possesses three properties – "namespace", "id", and "articleName" – and is designed to represent record identifiers ("ID numbers") from well-known resources (e.g. GO, EMBL). The value of the namespace property is a member of the Namespace Ontology, the value of the id property is the record-identifier within that resource, and the value of the articleName property indicates (as a human-readable phrase) the semantic nature of the relationship between a given class and a class that is in a HAS or HAS-A relationship to it. For example a visual representation of DNA Sequence data type is shown in Figure 11. The data type DNA sequence inherits an Object which is a String from Generic Sequence data type and an Integer from Virtual Sequence data type.

The MOBY Namespace Ontology is derived from the Cross-Reference Abbreviations List of the Gene Ontology project (http://geneontology.org/cgi-bin/xrefs.cgi). It is simply a list of abbreviations for the different types of identifiers that are used in bioinformatics. The combina-

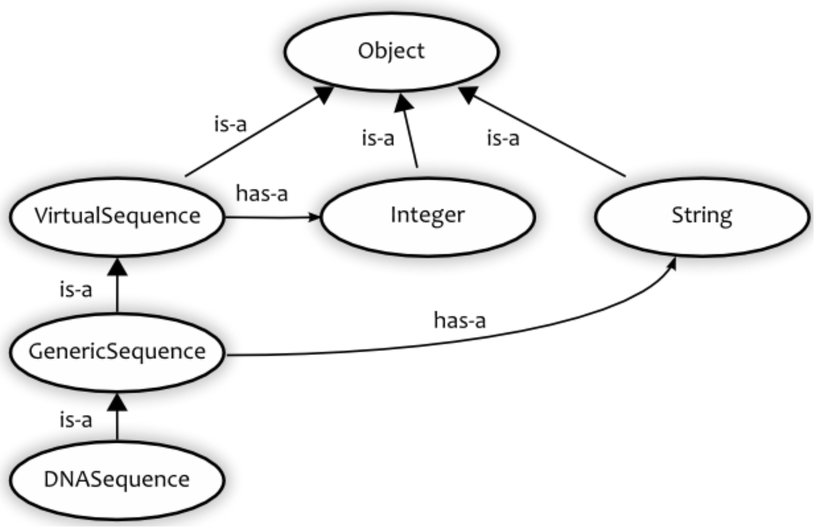


Figure 11 Sample of BioMOBY object ontology.

tion of a namespace and an id for a BioMOBY Object represents a unique identifier for a piece of data. For example, KEGG has KEGG_ID identifiers that are used to enumerate all of their sequence records and it is defined as "KEGG_ID" in the Namespace Ontology.

If compared with the RDF data model as shown in Figure 3 one can see that the BioMOBY object model depicted in Figure 10 is more or less the same. Nevertheless, the BioMOBY object model, although conceptually similar to RDF, uses a different serialization format. For example the syntactic representation of DNA sequence data type of Figure 11 in XML is the following.

```
<DNASequence namespace="NCBI_gi" id="111076">
   <Integer namespace="" id="" articleName="length">38</Integer>
   <String namespace="" id="" articleName="SequenceString">
      ATGATGATAGATAGAGGGCCCGGCGCGCGCGCGCGC…
   </String>
</DNASequence>
```

Basically each BioMOBY object is represented in XML with an element with the same name and all the HAS/HAS-A tionships of a BioMOBY object are serialized as “sub-elements” contained in the object’s XML element.

Besides the Object and Namespace tologies there’s also the Service tion ontology of BioMOBY. The Service Ontology (Figure 12) is an attempt to organize bioinformatics tools into a zation system, such that tools of similar functions are grouped together, and can be discovered by the biologist using a consistent naming system. It’s a simple subclass hierarchy which defines a set of data manipulation and bioinformatics analysis types. These include classes such as

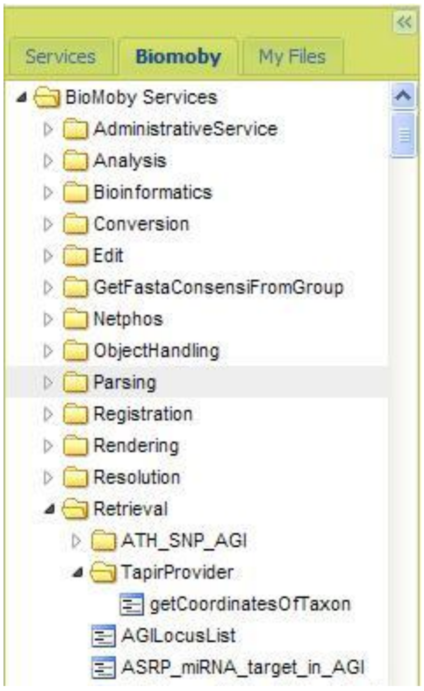


Figure 12 Part of the BioMOBY Service Ontology as shown in the ACGT Workflow Editor

- Retrieval for retrieval of records from a database
- Parsing for the extraction of information from various flat-file formats
- Conversion for data-type syntax changes.

Sub-classing is used to define more precise types of service operation.

3.5 Service Semantics in ACGT

Semantics provide “meaning” for “understanding” the entities and the processes in a domain of discourse. It is nevertheless true that defining the meaning of things as the main task of ontology engineering never ends. There is usually a multitude of views, abstraction layers, uses and goals to provide “meaning” to a certain artefact. Therefore we need to clearly define the role of semantics descriptions of services and to prioritize the different use cases of them in order to provide some useful and practical solution. For these reasons we have selected the service *discovery*, *selection*, and “*matchmaking*” (composition) as the primary use cases where semantics descriptions for services fit in. All of these are advanced features of a modern problem solving environment such as the Workflow Editor and Enactment environment that the ACGT, and in particular work package 9, aims to deliver.

Revisiting the different kinds of semantic descriptions that were described in the introduction of Section 3 and can be specified for the ACGT services it seems that the functional and informational descriptions are of particular importance. The Functional descriptions give semantics descriptions about the service capabilities and therefore are important for the discovery of services based on what they can do for the user. Also at the semantic level the Informational descriptions support the discovery, integration, and composition scenarios for web services since they provide information about the input and output messages of the services. On the other hand the Technical information is strictly at the syntactic level, specifying transport and communication specific features or requirements of the services. Finally Behavioural descriptions are an interesting case where especially the externally visible behaviour of the service can be used for automatically constructing parts of a workflow or "workflow templates".

Another aspect related to the technologies we have briefly described above is the level of the ontologies employed. We need to distinguish two types of ontologies:

- Foundational (upper-level) ontologies, such as the ones provided by OWL-S and WSMO. These are domain agnostic ontologies that aim to provide the general framework used for service annotation and discovery.
- Domain specific ontologies, such as the BioMOBY's data types. These ontologies provide some classification of domain specific terms and concepts and therefore are orthogonal to the upper-level ontologies. These ontologies can be used to support service discovery and also composition of services based on the annotation of inputs and outputs.

As an example an upper ontology for services can define that the proposition "a service offers some functionality" is something that can be expressed but a more ground, domain specific ontology, like the BioMOBY Service ontology, is needed in order to specify what the possible "functionalities" are.

As an upper ontology for the ACGT services we have chosen OWL-S and more specifically its ServiceProfile ontology. This ontology provides a bare minimum for supporting service discovery and selection:

- Service name and description use some human readable but also machine searchable text information
- The service parameters are represented along with their categorization to inputs and outputs
- The classification of services based on their capabilities and functionality can be supported by the building of a domain specific ontology with the ServiceProfile class itself as the root of the hierarchy

Additional strong features of the OWL-S such as its modelling of preconditions and effects or its Process ontology to specify how the service is composed out of other services, which corresponds to the Behavioural semantics descriptions, are not used. These admittedly useful features are either not supported by the existing infrastructure of the ACGT, e.g. the Metadata Repository, or are not too important in supporting the service discovery use case. On the other hand WSMO although is distinguished for including the notion of mediators in the ontology specification appears to be more heavyweight and business oriented[5]. An additional

[5] There is an undergoing effort to provide a simpler conceptual framework for Web Services called WSMO-Lite (http://cms-wg.sti2.org/TR/d11/v0.2/)

"drawback" is also the fact that WSMO is defined in terms of WSML whereas OWL-S is based on OWL. Also SAWSDL is mostly involved on how to link the semantics directly from inside the WSDL service descriptions, which is not of immediate use in the ACGT. This is because the service metadata descriptions are more easily accessible and indexed in the Metadata Repository. Also as criticized elsewhere it may be the case that referencing a single (or even a list of) concept identifiers is not enough for capturing the service semantics in the general case.

As described above an upper level service ontology like OWL-S is not enough: there should be also some domain specific ontology (or ontologies) that fill in the missing semantics. BioMOBY ontologies provide such domain specific ontologies. In particular its Object ontology supplies a large set of bioinformatics data types and formats that can be used for annotating the service parameters. Also the BioMOBY Services ontology accommodates a hierarchy of service capabilities that is again bioinformatics specific. Nevertheless, the choice of the domain specific ontologies is not very important for providing the semantic integration and interoperability checking because they are not defining factors for building a generic semantic architecture. This is in contrast with the foundational (upper-level) ontology which is needed in order to specify the ontology framework used for discovering services and checking their semantic compatibility. We next describe how such a semantic framework is designed for ACGT and the use of domain and foundational ontologies in it.

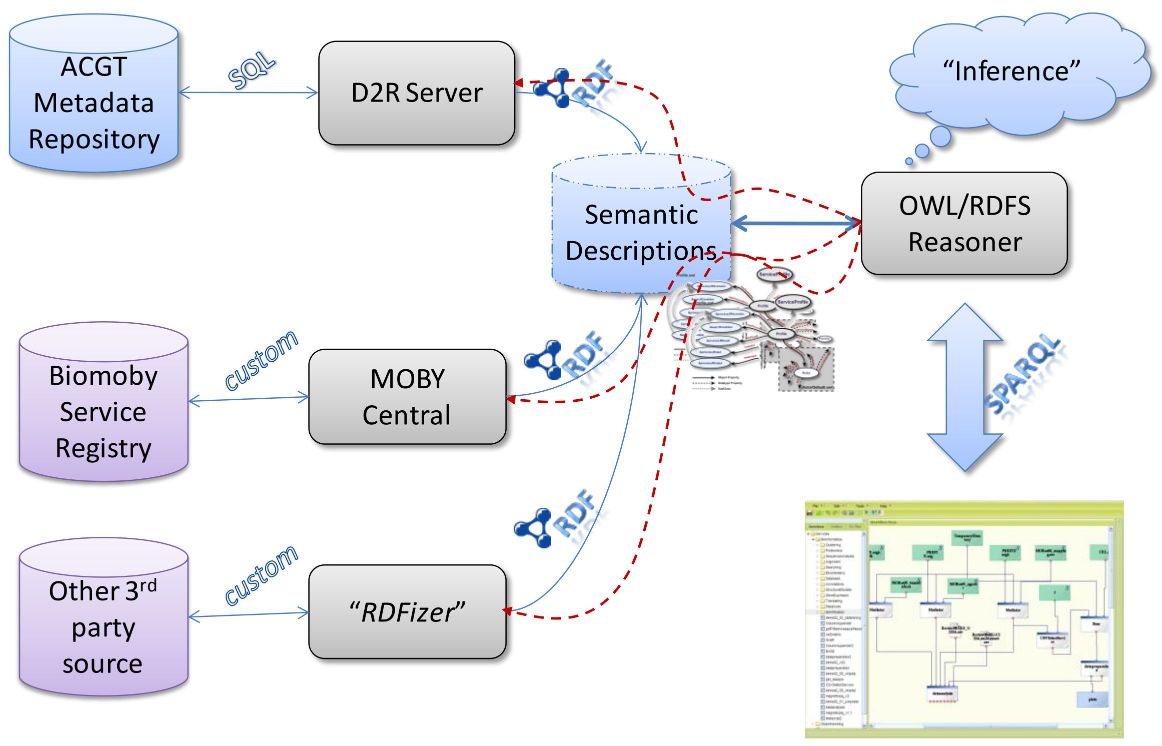


Figure 13 The Semantic Integration Framework

4 A Semantic Framework for the ACGT Services and Tools

The general architectural view of this framework is shown in Figure 13. It basically defines the following components:

- The service registries and repositories. In ACGT this is the Metadata Repository but additional third party registries exist, such as the BioMOBY ones. These are the primary sources of service descriptions and need not be implemented with the same technologies or contacted and searched with the same protocols. We assume though that conceptually they are compliant with the minimal upper level service ontology we have selected, which is the OWL-S Service Profile.
- The "RDFizers" are components (either "in-house" developments or "off the shelf") for exporting the service registries information in the RDF format and in the schema defined by the foundational ontology.
- The Reasoner is the component that performs the actual tasks of service discovery or matching by employing certain inference rules on the RDF data exported by the "RDFizers". These inference rules are of course in accordance with the foundational and domain specific ontologies.
- The interested user interface tools, like the workflow editor, or other services contact the Reasoner in order to make the proper entailments and inferences and answer their queries.

More detailed descriptions of these components follow.

4.1 Metadata Repository

In ACGT the Metadata Repository described in D6.3 is the authoritative source of information for discovering what the available ACGT services are, what they offer, what they need in order to be invoked, who provides them, etc.

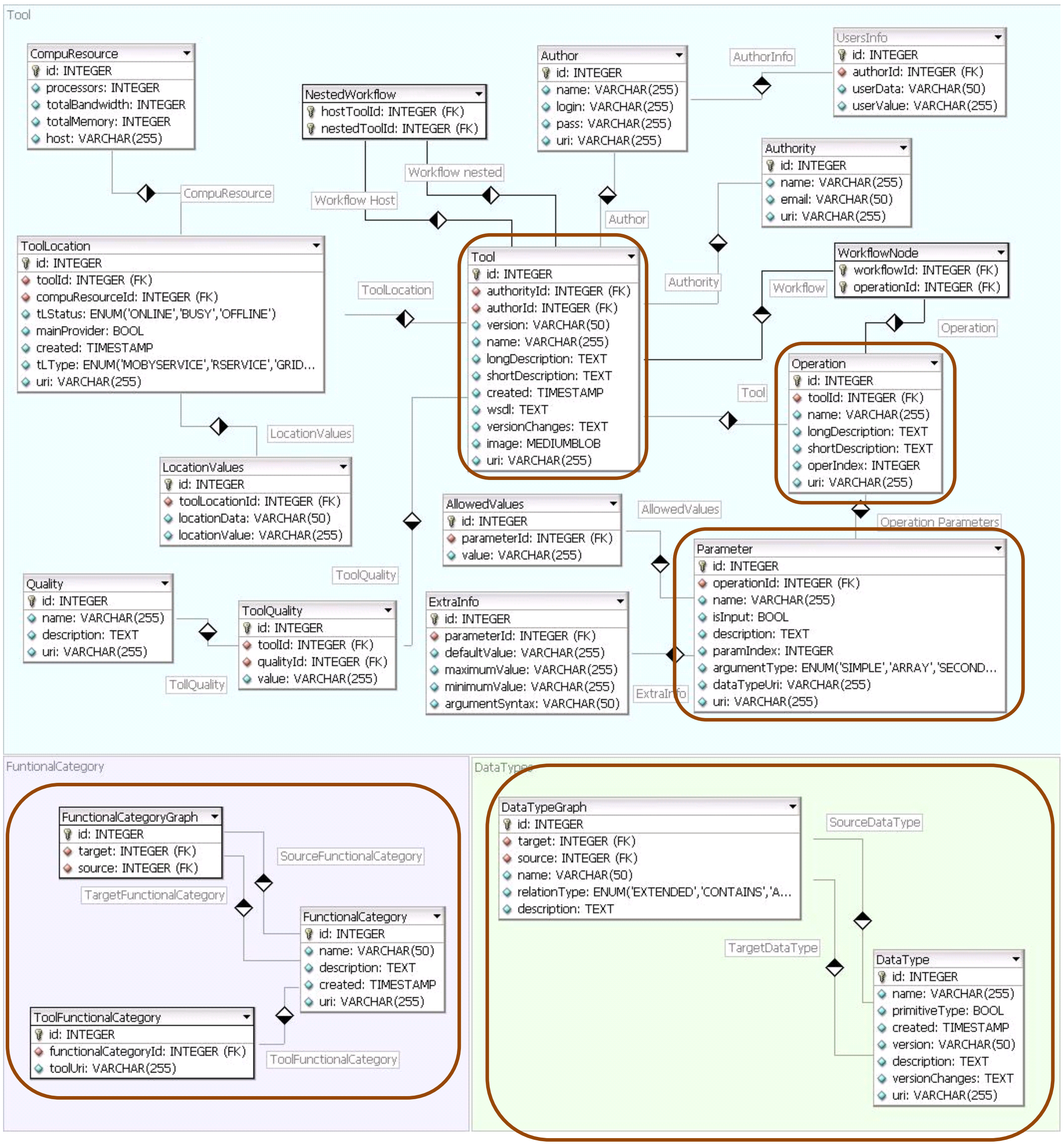


Figure 14 The ACGT Metadata Repository schema

The ACGT metadata repository is actually a relational database that stores the service related metadata according to schema shown in Figure 14. The main entities of the schema are services (tools), operations (specific functions of a tool), workflows (pre-defined flow of data between several operations), functional categories (descriptions of tool functionality) and data-types (the input or output data type of operation parameters).The repository itself is accessed through the Modular API (“mAPI”) by the various ACGT components. Nevertheless, in order to provide a semantic web compliant view of the repository’s contents a map-

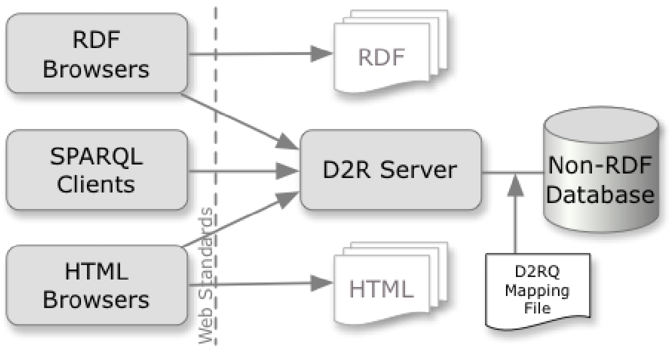


Figure 15 The D2R Server architecture

ping to RDF is needed. For this reason the D2R Server[6] is used. D2R Server (Bizer & Cyganiak, 2006) is a tool that permits the publication of the contents of relational databases and SPARQL compliant query interfaces of their contents (see Figure 15).

The way the relational data are transformed to RDF is specified in a mapping document that configures the D2R Server. The mapping format supported by the D2R Server is a declarative language to describe mappings between relational database schemas and either OWL or RDFS ontologies. The mappings allow RDF applications to access the contents of relational databases using Semantic Web query languages like SPARQL. Doing such a mapping requires us to choose how tables, columns, and values in the database map to URIs for classes, properties, instances, and data values. A specific mapping file has been created for the ACGT Metadata Repository that makes the data exported by the D2R Server compliant with the subset of the OWL-S Profile ontology we have specified above. The most important transformations guided by this mapping file are the following:

- A "Tool" entry and the corresponding "Operation" entry in the Metadata Repository represent a Service so they are mapped to an owls:ServiceProfile instance.
- A "Parameter" entry designates an input or output parameter and therefore maps to the corresponding instance that is the object of the owls:hasInput or owls:hasOutput property.
- The "FunctionalCategory" and the "FunctionalCategoryGraph" tables build up a classification of service functionalities where each service functionality is an indirect (or direct if it's the "root") "child" of the owls:ServiceProfile class though the rdfs:subClassOf property.
- Similarly the "DataType" and "DataTypeGraph" tables build up a domain specific ontology (actually hierarchy) of (semantic) data types for the annotation of input and outputs.

The D2R Server instance for the ACGT Metadata Repository can be accessed at http://iapetus.ics.forth.gr/mrepo/

4.2 BioMOBY Service Registry

The BioMoby architecture is based on the consept of a central registry ("MOBY Central") where the services descriptions are stored. Currently the most comprehensive BioMOBY registry seems to be that at University of Calgary, Canada (http://moby.ucalgary.ca/). The contents of this registry are publically available and can be retrieved in RDF format:

[6] http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/

Object ontology	http://moby.ucalgary.ca/RESOURCES/MOBY-S/Objects
Service Ontology	http://moby.ucalgary.ca/RESOURCES/MOBY-S/Services
Namespace Ontology	http://moby.ucalgary.ca/RESOURCES/MOBY-S/Namespaces
Services (instances)	http://moby.ucalgary.ca/RESOURCES/MOBY-S/ServiceInstances

Table 1 Web addresses for accessing the BioMOBY service registry in Canada

The RDF service descriptions retrieved are using the myGrid/Moby Service Ontology (Wolstencroft et. al, 2007) that is summarized in Figure 16.

The core entity in this service ontology model is the operation, which represents a unit of functionality for the user. Operations could be grouped into units of publication represented by the Service entity. An Operation has input and output parameters. In turn, each input and output parameter has a name, a description and belongs to a certain namespace denoting its semantic domain type.

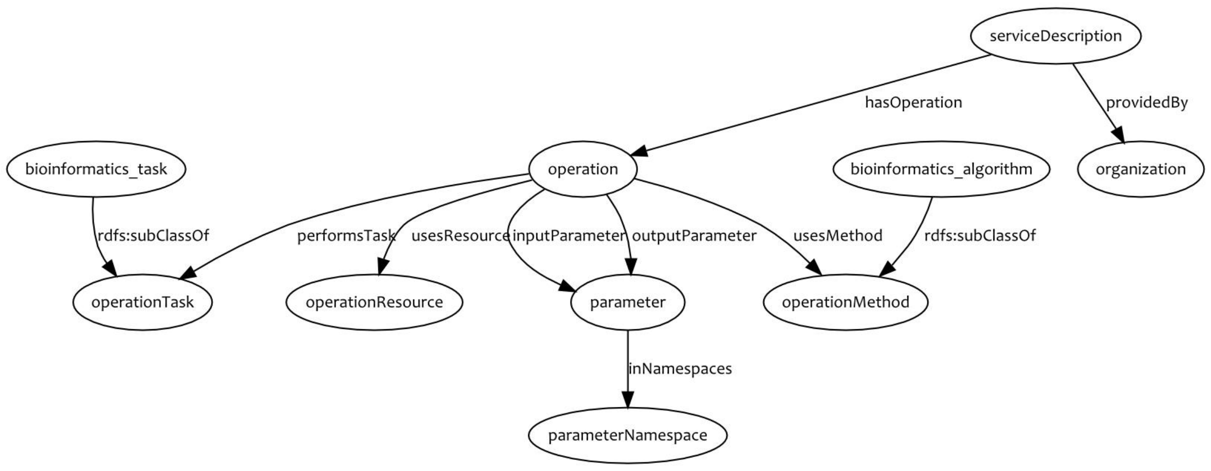


Figure 16 myGrid-Moby Ontology

The scope of the ontology is limited to support service discovery. Each hierarchy contains abstract concepts to describe the bioinformatics domain at a high level of abstraction. By combining the terms from the ontology, descriptions of services are constructed to detail:

- What the service does
- What data sources it accesses
- What each of the inputs and outputs should be
- Which domain specific methods the analysis involves

Combining the domain ontology and the service ontology enables full descriptions of services. To take an example the BLASTn service would be described in the following way:

- The overall task being performed by the operation (i.e. the biological operation it performs): aligning
- The bioinformatics algorithm used (i.e. the underlying scientific method): NCBIBlast
- The data resource it accesses: NCBI GenBank database
- The number of inputs: 1
- The number of outputs: 1

- Input 1: DNA sequence (fasta format)
- Output: Blast report

Integration in the general framework presented above requires the aligning of this ontology with the OWL-S Service Profile. This can be easily achieved by providing additional axioms to the Reasoner that allow the merger of the two ontologies:

- The BioMOBY operation class corresponds to the OWL-S service profile, because as described above it's a unit of functionality offered to the user. Therefore a statement like this moby:operation rdfs:subClassOf owls:ServiceProfile will provide the Reasoner with enough knowledge for linking the two ontologies and supporting inference across both of them.
- The inputParameter relation corresponds to the hasInput property in OWL-S. This can be stated like this: moby:inputParameter rdfs:subPropertyOf owls:hasInput .
- Similarly the outputParameter relation corresponds to the hasOutput property in OWL-S. This can be stated like this: moby:outputParameter rdfs:subPropertyOf owls:hasOutput .

4.3 Reasoning and inference

Reasoning is performed for matching the description of services requests coming from the end user applications to the contents of the service knowledge base. The way this matching query to the semantic descriptions of the available services is performed is guided by the foundational and domain specific ontologies.

Inference is needed because these ontologies have specific "recipes" for extracting new knowledge. A simple example to make it clearer is the following:

Find me services that perform gene expression analysis

In this case "gene expression analysis" is a domain specific concept coming from the BioMOBY service ontology where it is identified as the moby:GeneExpressionAnalysis class. According to the way the OWL-S Service Profile ontology is used the answer to this request will be returned in variable ?s when the following SPARQL query is submitted:

```
SELECT ?s
WHERE {
        ?s a moby:GeneExpressionAnalysis
}
```

Even in this case inference is important and it is based on the fact that the BioMOBY Service ontology is a hierarchy of service capabilities with rdfs:subClassOf properties and the following rule is supported by all RDFS/OWL Reasoners:

```
IF ?x a ?c1 AND ?c1 rdfs:subClassOf ?c2 THEN ?x a ?c2
```

This means that the reasoner will return not only services that have been asserted to be in the specific category (class) but also services that belong to all its subcategories since according to the inference rule

In RDFS and OWL a distinction between an Individual and a Class is made. A class is generally defined as a set of individuals or the instances that belong to this class.

The participation of an individual to a class is stated with the "rdf:type" property that in serialization formats like Turtle can be abbreviated simply as "a". On the other hand classes can be put into hierarchies using the rdfs:subClassOf property which denotes subset relationships (since classes are sets).

The following uses these relations in a typical example:

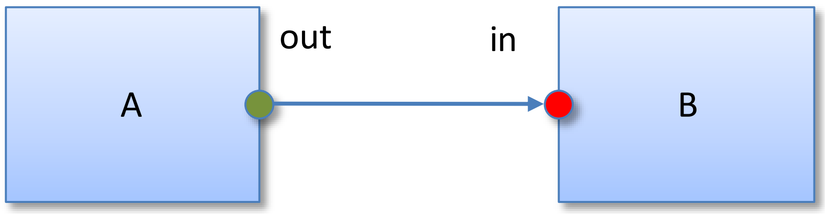
```
:Man rdfs:subClassOf :Mortal .
:Socrates a :Man .
```

The semantics of these relations allow for the following inference to be made:

```
:Socrates a :Mortal .
```

above the instances of the child categories belong also to the parent category.

Validation of input and output connections in a workflow is another example. As shown in the next image the "out" parameter of service A needs to semantically conform to the "in" parameter of service B in order to have a connection between the two services.



This validation of connections can be performed by checking the semantic types of the parameters. In particular if the semantic type of the "out" parameter is $\mathcal{T}_{out}$ and the type of the "in" parameter is $\mathcal{T}_{in}$, the connection is valid only if $\mathcal{T}_{out}$ is a subclass of $\mathcal{T}_{in}$, denoted as $\mathcal{T}_{out} \subseteq \mathcal{T}in$. This "subsumption" relation is needed so as to guarantee that the every instance (data) of the "out" parameter is compatible with the type of data the "in" parameter.

The RDF Schema standard defines the rdfs:subClassOf property for building hierarchies with the following inference rule

IF ?c1 rdfs:subClassOf ?c2 AND ?c2 rdfs:subClassOf ?c3 THEN ?c1 rdfs:subClassOf ?c3

By employing the rule above a reasoner can deduce that the "out" parameter need not have a direct subcategory of the class of the "in" parameter but any direct or indirect subclass of it.

Searching for services accepting a given data type is also a frequent use case. Assuming that the user has some data of type ?dt, she wants to find services that accept this kind of data as input. A possible SPARQL query to do that is the following

```
SELECT ?s
WHERE {
        ?s a owls:ServiceProfile ; owls:hasInput ?in .
        ?in a ?dt .
}
```

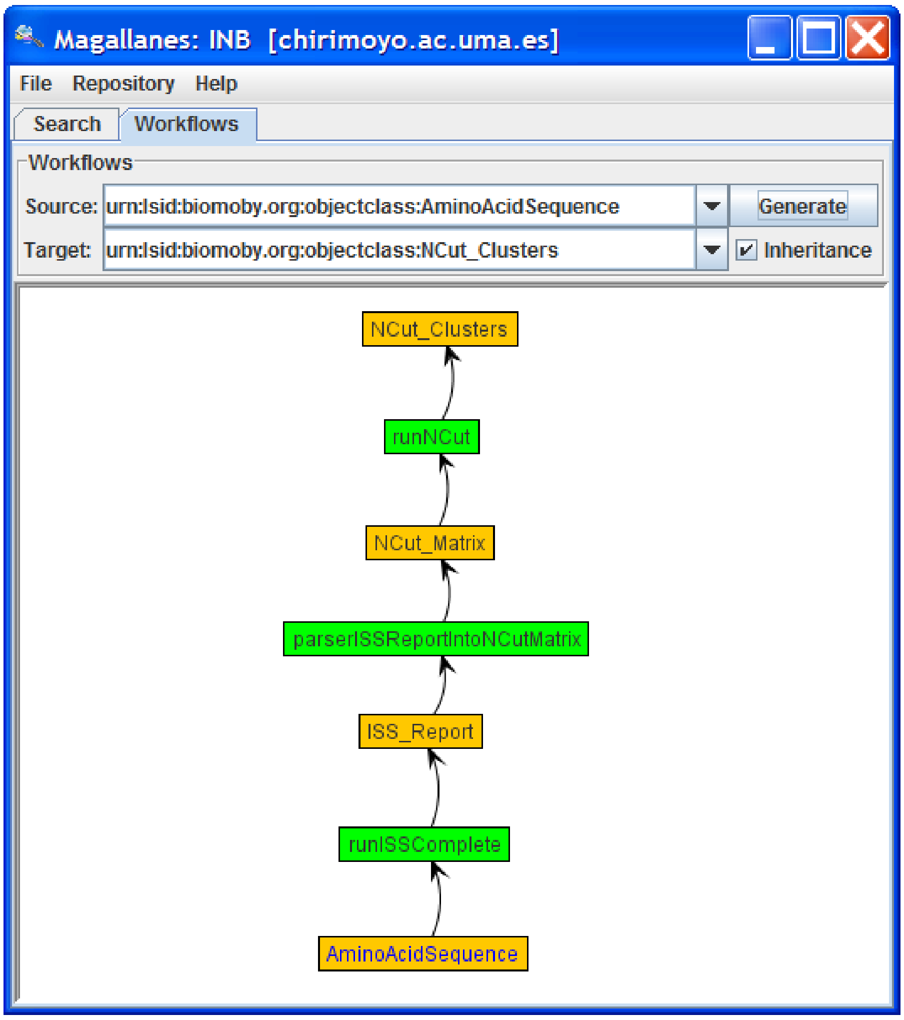
Since the operation class of BioMOBY has been defined to be subclass of OWL-S Service Profile and the moby:inputParameter is a rdfs:subPropertyOf owls:hasInput, this query will search also for BioMOBY services. The inference rules used are the transitive properties of rdfs:subClassOf (shown and used above) and rdfs:subPropertyOf :

IF ?x ?op1 ?y AND ?op1 rdfs: subPropertyOf ?op2 THEN ?x ?op2 ?y

Of course all these cases can be combined into more complex queries. A possible advanced and complex use case of the service discovery and composition tasks is presented in the next sub section.

4.4 Magallanes: a tool for automatic workflow creation

Bioinformatics research often involves combining independent web-services as workflows in order to solve more complex tasks. However, manual workflow creation is potentially complex and prone to errors. This represents a real challenge to life scientists who wish to utilize web-services.



Source: urn:lsid:biomoby.org:objectclass:AminoAcidSequence

Target: urn:lsid:biomoby.org:objectclass:NCut_Clusters

[✓] Inheritance

Figure 17 Magallanes client tool

Automatic workflow generation (or service composition) aims to automate the task of connecting independent services as much as possible. Two services can be connected if the output of one is compatible with the input of the other. The goal is to find an optimal set of services that match inputs with outputs.

Magallanes (Magellan) is a versatile, platform-independent Java library of algorithms that can be used for discovery of web services and associated data types. Additionally, Magallanes can connect compatible web-services into pipelining workflows that can process data sequentially to reach a desired output. In Figure 17, a standalone client using the Magallanes library is shown. User has located the AminoAcidSequence and NCut_Clusters data types and selected those as input and output data types. Magallanes used service metadata to create the workflow using three different services to complete the workflow.

The workflow generation uses the target data type and considers the available services which can be used to produce data of this data type and then connects services until it reaches the desired source data type. When several paths are possible, the user interface asks the user which alternative path to choose. Feedback is given to the user in the form of the service descriptions. After this stage, the workflow can be exported for further editing. Currently, such exports are possible for the Taverna workbench but the ACGT workflow editor format will be supported in future versions. Selection of alternative paths will be further supported by use of service quality information (such as availability rate, frequency of use; i.e. popularity).

The Magallanes library is directly compatible with the ACGT metadata repository through the use of the Modular API (see D6.3). It is therefore not implemented using the ontology based frawework presented above and the relevant Semantic Web technologies. Nevertheless it shows a desired end user functionality that can greatly facilitate the construction of scientific workflows.

Magallanes is further described in Deliverable 6.4 where the initial task of locating data types is discussed.

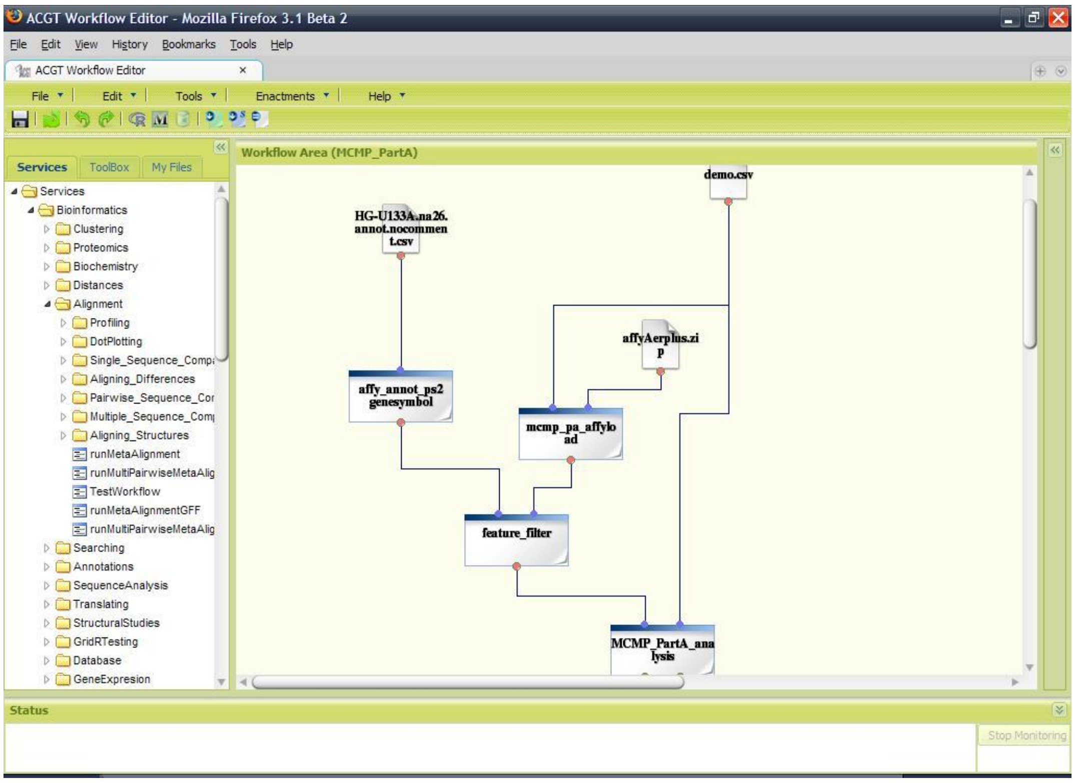


Figure 18 The ACGT workflow editor

5 Conclusions

The interoperability of the ACGT components is tested by the developers but it's also continually exercised by the users themselves. The ACGT Workflow Editor (Figure 18) is the end user application for the designing and execution of high level scientific workflows. In this web based application the users are facilitated to graphically combine the data retrieval and discovery services and the knowledge extraction and data analysis tools. The definition of the syntactic representation of the data and most importantly the annotation of the services with semantic metadata descriptions gives a lot of flexibility in the workflow editor for supporting user friendliness and intelligence. If properly annotated, incompatible services cannot be directly connected because the data types of their inputs and outputs do not conform to each other, either in the syntactic or the semantic level, while service recommendation and intelligent workflow composition can be also supported.

Therefore the semantic annotation of data and services is of utmost importance. In this document a generic Semantic Web compliant framework has been describe to support the semantic integration of the ACGT services and tools but also additional services outside the ACGT domain. We have argued that the Semantic Web provides the necessary tools to have such generic integration architecture implemented and deployed. Technologies and standards in this domain are an active area of research and in this deliverable a representative subset of them has been presented. Nevertheless the needs in the ACGT are more in the area of service discovery and “match making” and therefore a simple subset of the OWL-S upper ontology has been selected to be used.

Upper ontologies address only a small part of the problem: the bulk of work lies in developing industrial scale ontologies that capture the real domain semantics. To this end BioMOBY Object and Service ontologies present a possible solution that can be reused. It is not the objective in the work presented here to strictly define what is the specific domain ontology that should be used throughout the ACGT platform. Instead the general architecture for the semantic integration is presented, which should be fully implemented in the near future.

References

Beckett, D., Berners-Lee, T. (2008). Turtle - Terse RDF Triple Language, *W3C Team Submission*, Latest version available at http://www.w3.org/TeamSubmission/turtle/.

Berners-Lee, T. & Hendler, J. & Lassila, O. (2001). The Semantic Web, *Scientific American*, 284(5), 28-37

Berners-Lee, T., Bray, T., Connolly, D., Cotton, P., Fielding, R., Jeckle, M., et al. (2004). Achitecture of the World Wide Web, Volume One. *W3C*, Retrieved 15 June 2008, from http://www.w3.org/TR/webarch/

Bizer, C. and Cyganiak, R. (2006) "D2R Server: Publishing Relational Databases on the Semantic Web", 5th International Semantic Web Conference

Bizer, C., Heath, T., Idehen, K., & Berners-Lee, T. (2008). Linked data on the web (ldow2008). In *WWW '08: Proceeding of the 17th international conference on World Wide Web* (pp. 1265-1266). New York, NY, USA: ACM.

Brickley, D., & Guha, R. (2004). RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004. *World Wide Web Consortium*.

Dean, M., Schreiber, G., et al. (2004). OWL Web Ontology Language Reference. *W3C Recommendation, 10.*

Deliverable 6.3 "Demonstration and report of a repository of knowledge-discovery-related metadata"

Deliverable 6.4 "The integrated ACGT analysis environment"

Kopecký, J., Vitvar, T., Bournez, C., and Farrell, J. (2007) "*SAWSDL: Semantic Annotations for WSDL and XML Schema*." IEEE Internet Computing 11(6), pp 60-67

McIlraith, Sheila A., Cao Son, Tran, & Zeng, Honglei (2001). Semantic Web Services, *IEEE Intelligent Systems*, 16(2), 46-53

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., and Sycara, K. (2004) "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. http://www.cs.cmu.edu/People/softagents/papers/OWL-S-SWSWPC2004-final.pdf

Prudhommeaux, E., & Seaborne, A. (2008). *SPARQL Query Language for RDF. W3C Recommendation 15 January 2008*. Available from http://www.w3.org/TR/rdf-sparql-query/

Lassila, O., Swick, R., et al. (1999). Resource Description Framework (RDF) Model and Syntax Specification. *W3C Recommendation, 22*, 2004-03.

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005) "Web Service Modeling Ontology" Applied Ontology, 1(1):77–106

Sheth, A. (2003) “Semantic Web Process Lifetime: Role of Semantics in Annotation, Discovery, Composition, and Orchestration.” Workshop on E-Services and the Semantic Web, WWW 2003.

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, 11 (2).

Wilkinson M.D. et. al (2008) “Interoperability with Moby 1.0 - it's better than sharing your toothbrush!“, Brief Bioinform. 2008 May;9(3):220-31

Wolstencroft, K., Alper, P., Hull, D., Wroe, C., Lord, P.W., Stevens, R.D., and Goble, C.A (2007) “The myGrid ontology: bioinformatics service discovery,” International Journal of Bioinformatics Research and Applications (IJBRA), 3(3), 303-325