Managing Semantic Metadata for the Semantic Grid

Chen L.[1], Shadbolt N.R.[1], Tao F.[1], Goble C.[2], Puleston C.[2], Cox S.J.[3]
[1] *Department of Electronics and Computer Science*
University of Southampton, Highfield, Southampton, SO17 1BJ, U.K.
{lc, nrs, ft}@ecs.soton.ac.uk
[2] *Department of Computer Science*
University of Manchester, Oxford Road, Manchester, M13 9PL, U.K.
{carole, colin.puleston}@cs.man.ac.uk
[3] *School of Engineering Sciences*
University of Southampton, Highfield, Southampton, SO17 1BJ, U.K.
sc@ecs.soton.ac.uk

Abstract

Research on the Semantic Web and Web/Grid resource description, discovery and composition is booming but there is currently little effort on a systematic and integrated approach to the management of resources' Semantic Metadata (SMD), nor on key tools that add, store and reuse SMD. In this paper we propose a generic framework for managing resource SMD, in which ontologies are used for metadata modeling and the Web Ontology Language (OWL) for semantic representation. Generated resource SMD are archived in a knowledge repository enhanced with Description Logic (DL) based reasoning capability. A raft of tools, mechanisms and APIs are developed to support SMD management lifecycle, including metadata generation, semantic annotation, knowledge storage and semantic reuse. Both the framework and its supporting technologies have been applied to a large existing e-Science project, which has produced a working resource management prototype. While SMD can be exploited in many ways with regards to resource discovery, provenance and trust, we illustrate their usage through a knowledge advisor that assists resource assembly and configuration in the context of engineering design search and optimisation.

1. Introduction

The success of Grid computing [1], i.e. flexible, secure, coordinated resource sharing among dynamic collections of individuals and institutions, relies on the effective discovery and seamless aggregation of required resources on the Grid. To discover and use the "right" resources for the "right" problem is not a trivial job. Users need to use not only resources' functionality information but also their own selection policies with regards to reliability, invocation cost, provenance, quality of service etc, to determine the resources they prefer to utilise. All such information should be precisely and consistently derived from resources' original descriptions. Consequently this requires resource providers to augment their resource descriptions with additional information, i.e. metadata. Using metadata well-informed decisions can be made about data and processes based on logical inferences. Metadata is usually intended for consumption and interpretation by machines, rather than by humans.

Metadata exists at all levels of the Grid from low level core Grid services to high level application resources. For example, the Globus Monitoring and Discovery Service (MDS) [2] is a core Grid metadata service that stores information about Grid resources and their status. In the service-oriented Grid computing paradigm [3] application-level metadata are associated with Web/Grid services. There are two categories of metadata about a resource. One describes resource functionality, i.e. metadata about resource capabilities and interfaces. The other describes non-functionality properties and/or attributes, i.e. metadata about provenance, performance, security and access policies, and so on.

Metadata have been used for Web/Grid service publishing and discovery [4]. For instance, a UDDI repository stores and represents metadata about the capabilities and interfaces of services using XML[1] [5]. However, XML, the representation language for service description, has a weak data model incapable of capturing genuine semantics, relationships or constraints. While it is possible to extend XML-based service descriptions to incorporate rich metadata, XML fails to support

[1] XML, RDF are all W3C standards, see www.w3.org for details

automated interoperability without necessitating human intervention. Furthermore, the types of metadata required for describing resources will naturally vary greatly between individuals, organisations, and scientific communities, using comments as metadata will not bridge the gap of interoperability. To make resource providers and consumers understand each other, an abstract and highly flexible conceptual representation of metadata, i.e. a metadata model, is required for service descriptions in a problem domain.

More recently advanced knowledge technologies [6] including semantic web technologies [7] have been used to bridge the divide between current endeavour and the vision of Grid computing with the ultimate purpose of the realisation of semantic Grid [8]. The inclusion and use of semantic annotations in Web/Grid resource publishing promise to make Web/Grid-based information and services both accessible and understandable to agents and other applications. There is currently vigorous research into individual areas such as semantic service description [9] [10], discovery [11] and composition [12] [13]. However, there is little effort towards a systematic and integrated approach to managing resource SMD, i.e. to streamline the process of generation, archiving, manipulation, retrieval and use of semantic metadata. There is also little experience on key tools such as those that add, enrich, store and search SMD.

In our work we propose a SMD management framework that focuses on the generation, storage and utilisation of resources' SMD. The framework harvests well-developed individual Semantic Web technologies. It is intended to be applicable to any real world domains. We also take a broad view on Web/Grid resource representation, which means the framework can manage Web/Grid resources not only in the web service format but also any representational models. In developing the resources' SMD management framework we have addressed the following issues:

- Data/metadata modeling using upper-level resource ontologies and domain-specific ontologies;
- Semantic annotation for data/metadata, large scale semantic information storage and developing tools that support these activities;
- Advanced reasoning and query mechanisms against underlying knowledge repository and tools and/APIs that support these activities;
- The exploitation of the management system in a real Grid application.

In tackling the above problems we have made the following contributions:

- We describe and design a generic framework for managing resources' SMD;
- We develop a resource description and registration wizard to annotate and archive the semantic metadata of Grid resources within a knowledge repository;
- We develop a centralised knowledge repository for the publishing and deployment of Grid resources, which supports a uniform way of resource discovery and retrieval;
- We provide a DL-based query interface and a semantic web based knowledge advisor to facilitate resource reuse by making use of metadata and semantic descriptions;
- Components of the SMD management system have been successfully used or integrated in a real e-Science project - Grid enabled optimisation and design search in engineering [14].

The paper is organised as follows. In section 2 we introduce a resource's SMD management framework addressing the main issues of management lifecycle. Section 3 describes individual components, their underlying technologies and corresponding implementations in GEODISE. In section 4 we present an example application of the SMD in the context of engineering design and optimisation. We conclude the paper in section 5.

2. Resource's SMD Management Framework

Traditionally resources are generated by resource users for their own consumption. Here resources refer to assets such as software packages and databases, capabilities such as computational algorithms, and knowledge. These resources are usually either discarded or archived somewhere after use. Stored resources are in most cases only accessible and consumable by resource creators. The resource lifecycle is short and limited to the resource creator/user. Grid computing, which is all about resource reuse and sharing, has significantly changed the scope and expectancy of a resource lifecycle. It brings up the issue of resource management, i.e. resource generation, publishing, storage, discovery, reuse and maintenance on the Grid.

We have developed a generic framework for SMD management for Web/Grid resources as depicted in Figure 1. The framework focuses on the management of high level application resources rather than low level core Grid services. But we take a broad view of Grid resource representations. They could be web services and any other forms such as computational algorithms and functions. The framework is designed based on the state of the art of semantic web technologies. First, we use ontologies as the conceptual backbone that supports the whole lifecycle of resource management, including metadata modeling, resource publishing, archiving, discovery and seamless use. Ontologies are built based on the domain knowledge of distributed Web/Grid resources as shown in Figure 1. An ontology-based metadata model will capture the required concepts and their relations needed for resource descriptions. It will allow for flexibility and adaptation to accommodate diverse metadata and future changes within the field. Resource descriptions with rich SMD provide more possibility for flexible and seamless resource reuse. For example, semantic metadata could be used by a knowledge advisor to semantically match two resources and determine whether they are compatible to each other in term of workflow composition. Autonomous software agents such as resource brokering agents can use the information to mediate resource interactions or schedule a resource-based application. These agents would have access to ontologies and inference engines to support their decision-making processes.

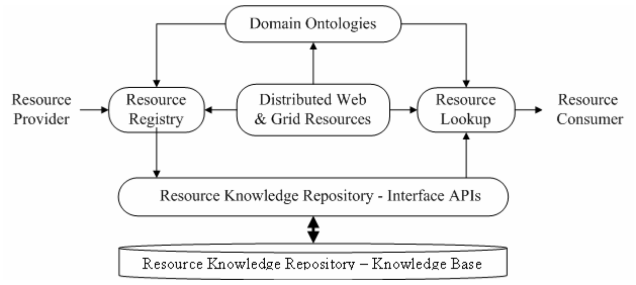

Domain Ontologies
Resource Provider
Resource Registry
Distributed Web & Grid Resources
Resource Lookup
Resource Consumer
Resource Knowledge Repository - Interface APIs
Resource Knowledge Repository – Knowledge Base

Figure 1. The generic management framework

Second, we use OWL, the standard web ontology language, as the resource description formalism to describe both resource data and metadata. Therefore, all information associated with a resource is uniformly represented and coded into one resource knowledge repository. Compared to using XML for resource data/metadata representation and RDF/OWL for semantic information representation, this approach avoids the overhead for managing the interaction and consistency between a UDDI service directory and a semantic data repository. OWL-based resource descriptions enable a uniform way of searching for resources, i.e. basic searches can be performed based on key words and more sophisticated searches based on metadata and ontology-driven queries such as looking for more general or specialized examples of a service.

One key component of the framework is the Resource Registry. It is responsible for adding metadata and attaching semantic information to resource descriptions. The Resource Registry will also publish and archive the resource's semantic descriptions in the Resource Knowledge Repository. The Resource Knowledge Repository is similar to the UDDI in functionality but consists of rich metadata and semantic information. The framework supports a centralised resource knowledge repository. Grid resources will remain where they are on the Grid whereas their descriptions will be published in the knowledge repository. The knowledge repository contains data and SMD as well as the underlying ontologies related to the resource domain. A centralised resource knowledge repository facilitates resource discovery even if all providers are not online. It also gives opportunities to clients to offload the resource searching process to the resource knowledge repository servers, thus reducing communication costs.

The Resource Knowledge Repository has two components, the Interface API and Knowledge Base. All resource descriptions will be stored in the Knowledge Base that could be implemented as relational databases, triple stores [15] and/or instance stores [16]. The Interface API is used to (1) archive resource descriptions, and (2) reason against resource descriptions to retrieve appropriate resources (handlers to physical resources on the Grid).

The Resource Lookup is a service implemented as a lightweight standalone or browser-based front-end GUI. Its purpose is to facilitate users constructing various, syntactical and/or semantic, searches and queries by using ontology-driven forms and selection choices. These search criteria will pass on to underlying DL-based reasoning engines such as the FaCT reasoner [17] [18], which will discover the resources that meet the search criteria.

3. Managing Resource SMD in GEODISE

Grid enabled optimisation and design search in engineering [14] is one of the UK e-Science pilot projects. It is intended to enable engineers to carry out Engineering Design Search and Optimisation (EDSO) by seamless access to a state-of-the-art collection of optimisation and search tools, industrial strength geometry modeling and meshing tools, analysis codes and distributed computing and data resources on the Grid.

Engineering design search and optimisation has been practiced for decades. Huge amounts of expertise and algorithms are available in various formats. In GEODISE we use Matlab, a widely adopted engineering package in academia and industry, as the problem solving environment [19]. The main optimisation and search resources in Matlab are function scripts - a type of high-level computation programs that can accomplish various tasks in engineering design search and optimization by execution in Matlab environment. An example Matlab function can be found in the right-hand bottom panel of Figure 4 (Panel 6). It has been identified that the key issues to achieving GEODISE objective are (1) how to add rich metadata to these .m (Matlab) functions, (2) how to semantically enrich them, and (3) how to allow sophisticated reasoning and query capabilities over them.

We have applied the generic resource management framework to GEODISE. Figure 2 shows the design of the GEODISE resource SMD management system. The implementation of some of its main components is described below.

3.1 Ontologies

Ontologies play a central role in incorporating SMD into resource descriptions. An ontology is an explicit, shared specification of the various conceptualizations in a problem domain. It provides a common language not only for modelling metadata but also for adding meaning and relations to resource descriptions, thus facilitating semantic interoperability. Ontology representation languages, such as DAML+OIL (http://www.daml.org) and OWL, are built upon existing Web standards, such as XML and RDF Schema, and underpinned by description logic. Therefore, they support the classification of concepts based on their property descriptions - a description-based reasoning capability. Ontology-based metadata modelling enables semantics to be attached to resource metadata in an expressive manner, thus facilitating semantic search and query.

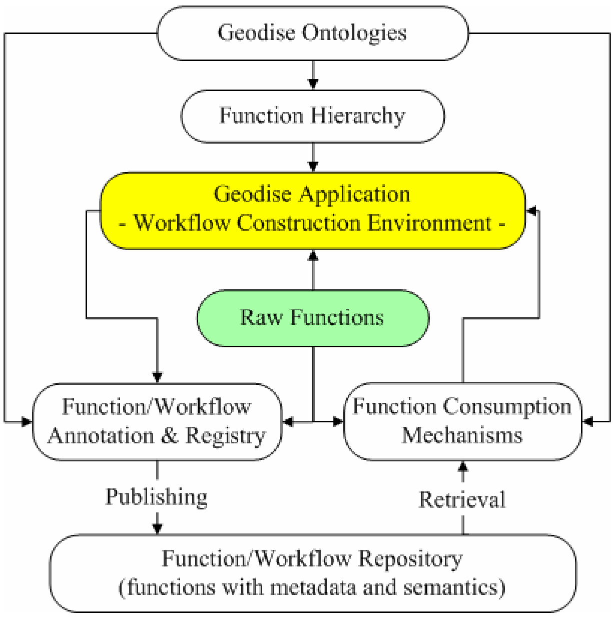
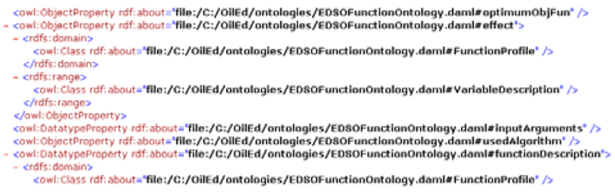

Geodise Ontologies
Function Hierarchy
Geodise Application
- Workflow Construction Environment -
Raw Functions
Function/Workflow Annotation & Registry
Function Consumption Mechanisms
Publishing
Retrieval
Function/Workflow Repository
(functions with metadata and semantics)

Figure 2. Function SMD management system

We have developed a function ontology for EDSO using classic knowledge engineering methods and the OilEd ontology editor [20]. The function ontology is based on the DAML-S [10] ontology in which we use function profiles to describe function metadata. Semantic descriptions are generated when linking resource metadata and interface (inputs/outputs) with underlying EDSO domain concepts. Figure 3 shows a fragment of the GEODISE function ontology in OWL.

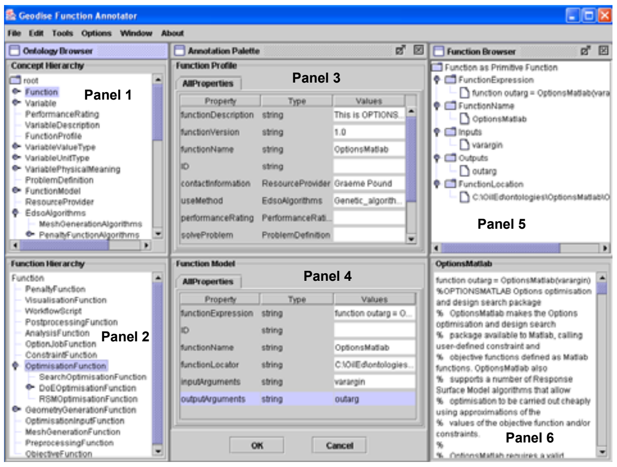
3.2 Function annotation and publishing

We have developed a tool, called Function Annotator [21], to accomplish the job of attaching SMD to function descriptions and publishing them into a resource repository. Figure 4 shows the GUI of the Function Annotator, which consists of an Ontology Browser, an Annotation Palette and a Function Browser. The Ontology Browser in the left-hand column contains a concept hierarchy (Panel 1) and a function hierarchy (Panel 2). The concept hierarchy presents the terms, relations and hierarchy of an ontology. It is used for users to browse and choose suitable concepts for descriptions. The function hierarchy displays available ontologically described functions under different function categories. These semantically enriched functions will be retrieved from a backend resource repository on a user's demand. They can be modified, edited and reused to generate new function descriptions.



```
  <owl:ObjectProperty rdf:about="file:/C:/OilEd/ontologies/EDSOFunctionOntology.daml#optimumObjFun" />
- <owl:ObjectProperty rdf:about="file:/C:/OilEd/ontologies/EDSOFunctionOntology.daml#effect">
  - <rdfs:domain>
      <owl:Class rdf:about="file:/C:/OilEd/ontologies/EDSOFunctionOntology.daml#FunctionProfile" />
    </rdfs:domain>
  - <rdfs:range>
      <owl:Class rdf:about="file:/C:/OilEd/ontologies/EDSOFunctionOntology.daml#VariableDescription" />
    </rdfs:range>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:about="file:/C:/OilEd/ontologies/EDSOFunctionOntology.daml#inputArguments" />
  <owl:ObjectProperty rdf:about="file:/C:/OilEd/ontologies/EDSOFunctionOntology.daml#usedAlgorithm" />
- <owl:DatatypeProperty rdf:about="file:/C:/OilEd/ontologies/EDSOFunctionOntology.daml#functionDescription">
  - <rdfs:domain>
      <owl:Class rdf:about="file:/C:/OilEd/ontologies/EDSOFunctionOntology.daml#FunctionProfile" />
```

Figure 3. Geodise function ontology



Property	Type	Values
functionDescription	string	This is OPTIONS...
functionVersion	string	1.0
functionName	string	OptionsMatlab
ID	string	
contactInformation	ResourceProvider	Graeme Pound
useMethod	EdsoAlgorithms	Genetic_algorith...
performanceRating	PerformanceRati...	
solveProblem	ProblemDefinition	

Property	Type	Values
functionExpression	string	function outarg = O...
ID	string	
functionName	string	OptionsMatlab
functionLocator	string	C:\OilEd\ontologies...
inputArguments	string	varargin
outputArguments	string	outarg

```
function outarg = OptionsMatlab(varargin)
%OPTIONSMATLAB Options optimisation
and design search package
%   OptionsMatlab makes the Options
optimisation and design search
%   package available to Matlab, calling
user-defined constraint and
%   objective functions defined as Matlab
functions. OptionsMatlab also
%   supports a number of Response
Surface Model algorithms that allow
%   optimisation to be carried out cheaply
using approximations of the
%   values of the objective function and/or
constraints.
%
%   OptionsMatlab requires a valid
```

Figure 4. Function annotator interface

The right-hand column of the GUI is the Function Browser, which is used to load Grid resources for semantic descriptions. As the main resources in GEODISE are Matlab functions, Function Annotator particularly targets them. We have provided a parsing capability to facilitate automatic information extraction based on the Matlab function interface and helper documentation. The extracted information, which includes function inputs/outputs as well as other metadata such as copyright, authors and summary, is listed in a tree structure in the top window of the right-hand panel (Panel 5). The bottom window (Panel 6) displays the source code of a Matlab function that gives users more flexibility for annotation. In particular, for compound functions such as Matlab scripts users can markup, copy and paste specific information from the source to be semantically annotated.

The Annotation Palette in the middle column of the GUI is where ontological description takes place. It consists of two panels, i.e. the Function Profile at the top (Panel 3) and the Function Model at the bottom (Panel 4). Function Profile contains two types of metadata. One is about function metadata such as what a function does, what it requires from and provides for users as well as information about authors, version, used methods, required preconditions etc. These metadata are specified using formal ontological concepts. The second type of metadata is about function input/output interface such as type and default values. Function Model is used to hold information on how a function works and how it can be invoked. This includes input/output arguments, location and expression signatures. For scripts it could contain information on embedded functions as well as their sequential details.

Users can enter this information by filling in ontology-driven forms and/or by selecting appropriate concepts from specific areas of the ontology (presented as either lists or sub-hierarchies), and where appropriate, specifying concrete values. The resulting input will be combined with some additional automatically generated data, (creator-name, date-of-creation, instance-ID, etc.), and used to create an instance in the function archive.

Using the Function Annotator a resource can be described at multiple levels of abstraction. For example an input could be specified as a high-level concept/type, or an instantiated object or concrete values. Resource providers can expose resources by adding information to the degree necessary.

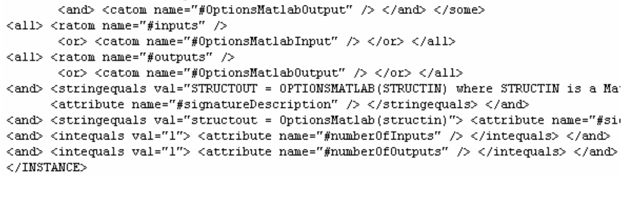
3.3 The knowledge repository

There are three different mechanisms to store annotations about a resource. First it can be added into the original resource with an embedded set of descriptions. Some annotation tools in the Semantic Web community uses this approach to attach semantics to web pages such as the OntoMat-annotizer (http://km.aifb.uni-arlsruhe.de/annotation/ontomat.html). Second, annotations can be saved in a separate file in the same location as the resource. Third annotations can be archived in a centralised knowledge repository separate from resources. Globus MDS has adopted this approach. In the context of Grid computing it is supposed that resources are owned by and geographically located in dynamic virtual organisations. These resources are published with explicit expressive descriptions exposing as much information as possible, so that they can be discovered, shared and reused. From this perspective we have decided to build a central knowledge repository for distributed resource SMD management, which can also serve as a registry service similar to the UDDI registry but with rich SMD.

When Grid resources are modelled using ontologies and represented in OWL, their descriptions will be generated as OWL individuals that are independent of the original resource formats and/or providers. In such cases both ontologies and ontological annotations will be saved together in an ontology file, which can be viewed as a knowledge base. Applications can then consume semantic information by accessing the ontology file and carrying out DL-based reasoning over individuals. Unfortunately, existing technologies, either Racer's assertion reasoning [18] or FaCT's terminological reasoning [17] over pseudo-concepts, fail to scale to hundreds of thousands of the size of individuals that is usually required by real scientific applications on the Grid.

We have adopted the instance store [16] technology to tackle this problem. The instance store uses a relational database such as MySQL and Oracle as a permanent storage media and a DL-based reasoner to support reasoning. This means that assertions over individuals are stored in a database, together with information inferred using a DL-based reasoner over the position in the ontological taxonomy of their corresponding descriptions. The DL-based reasoner deals purely with terminological reasoning functionality. As terminologies are fairly restrictive there will be no size limitation problem. Furthermore, pure terminological reasoning will significantly reduce reasoning cost while maintaining soundness and completeness. Retrieving individuals is then a combination of query against the database and subsumption and classification requests to the reasoner.

SMD in the instance store are represented in DIG (http://potato.cs.man.ac.uk/dig/interface1.0.pdf) format embedded in a general XML-based representation. Figure 5 shows a fragment of DIG descriptions in the GEODISE instance store. This fragment presents OptionsMatlab_1 function instance. We have provided an API to convert OWL individuals to DIG instances and vice versus.



```
<INSTANCE>
<and> <stringequals val="OptionsMatlab_1"> <attribute name="ONTOVIEW_NAME" /> </str
<and> <catom name="#FunctionSignature" /> </and>
<some> <ratom name="#inputs" />
       <and> <catom name="#OptionsMatlabInput" /> </and> </some>
<some> <ratom name="#outputs" />
       <and> <catom name="#OptionsMatlabOutput" /> </and> </some>
<all> <ratom name="#inputs" />
       <or> <catom name="#OptionsMatlabInput" /> </or> </all>
<all> <ratom name="#outputs" />
       <or> <catom name="#OptionsMatlabOutput" /> </or> </all>
<and> <stringequals val="STRUCTOUT = OPTIONSMATLAB(STRUCTIN) where STRUCTIN is a Ma
      <attribute name="#signatureDescription" /> </stringequals> </and>
<and> <stringequals val="structout = OptionsMatlab(structin)"> <attribute name="#si
<and> <intequals val="1"> <attribute name="#numberOfInputs" /> </intequals> </and>
<and> <intequals val="1"> <attribute name="#numberOfOutputs" /> </intequals> </and>
</INSTANCE>
```

Fig. 5. A fragment of DIG instance representation

3.4 Function consumption

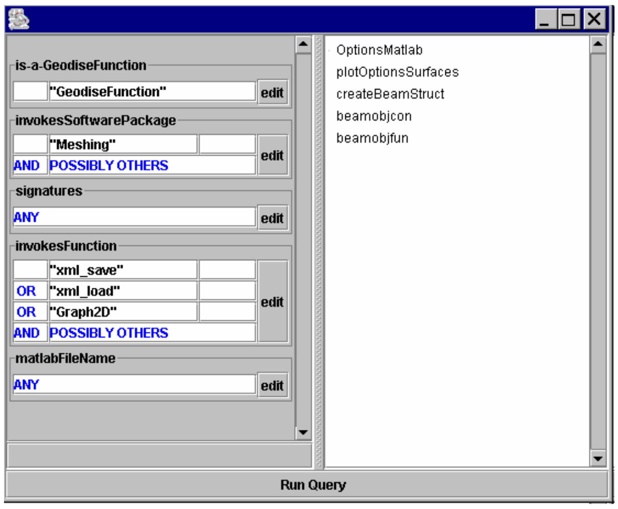
Once the SMD of functions and workflows are published into the Function/Workflow Knowledge Repository, these resources can be discovered and reused in more flexible ways. We have developed three main mechanisms for function/workflow reuse based on rich SMD.

The first is to retrieve functions/workflows in the repository in terms of semantic descriptions. Functions are classified into different categories when they are described using ontologies. By referencing the associated ontology users can obtain all functions under a specific function category (a concept and/or a property) and/or all functions in all categories. These functions can be presented in a hierarchical tree structure that shows their inter-relations and also facilitates selection.

The second mechanism we developed is ontology-driven SMD function discovery. This is particularly important for resource sharing on the Grid. Current Web/Grid service descriptions such as WSDL (http://www.w3.org/TR/wsdl) are usually more concerned with the signature of a service, i.e. the identifiers of the service and its parameters. Based on this description, it is usually impossible for software agents to figure out the precise meaning of service identifiers and functionality provided by the service. The lack of semantics in the abstract functionality description of the service, i.e. the capabilities of the service, makes it difficult for machines to both discover and use the service at the appropriate time.

The ontological description of functions allows the definition of classes of related functions and can establish links to other classes that describe specific function types and their properties. This makes function discovery much easier in terms of the built-in links, thus facilitating resource reuse. We have developed an ontology-powered query GUI, which can generate an ontology-driven query-building form, see Figure 6. The left-hand side panel of the form is used for building up the overall query expression. It consists of an ontology-driven template together with logical operators. To define specific query criteria for a particular property, users can click on the "edit" button corresponding to individual property. This will brings up a sub-window, see Figure 7. In this ontology-driven form the left-hand side panel shows the part of the query expression relating to that property (eg. invokesFunction), and allows the selection of different types of semantics for that sub-expression. The right-hand side panel of the form displays a hierarchy of concepts from the ontology and related instances from the instance store. Users can navigate the hierarchical structure to select appropriate attributes. Users can also create new instance-descriptions with arbitrarily complex semantic information and add them the sub-expression.

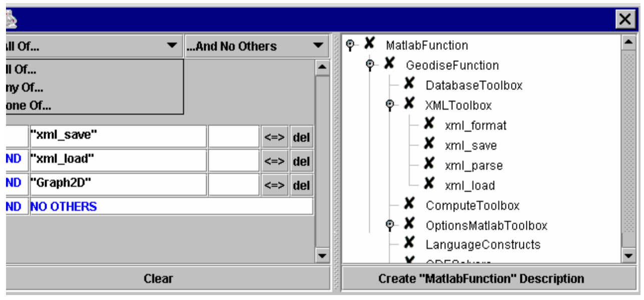
After a query expression is built up and the "Run Query" button is clicked. The underlying reasoning engines that support DL-based reasoning will reason against the instance store to obtain a set of entities matching all of the specified criteria. The results are displayed in the right-hand side panel as shown in Figure 6. Users can then make an appropriate selection to build new workflows.


is-a-GeodiseFunction
"GeodiseFunction" edit
invokesSoftwarePackage
"Meshing"
AND POSSIBLY OTHERS edit
signatures
ANY edit
invokesFunction
"xml_save"
OR "xml_load"
OR "Graph2D"
AND POSSIBLY OTHERS edit
matlabFileName
ANY edit

OptionsMatlab
plotOptionsSurfaces
createBeamStruct
beamobjcon
beamobjfun

Run Query

Figure 6. Top-level semantics-based query GUI


ll Of... ...And No Others
ll Of...
ny Of...
one Of...
"xml_save" <=> del
ND "xml_load" <=> del
ND "Graph2D" <=> del
ND NO OTHERS
Clear

MatlabFunction
GeodiseFunction
DatabaseToolbox
XMLToolbox
xml_format
xml_save
xml_parse
xml_load
ComputeToolbox
OptionsMatlabToolbox
LanguageConstructs
Create "MatlabFunction" Description

Figure 7. Property-related query construction form

The third mechanism to consume SMD is described in details in 4.2.

4. Using SMD in GEODISE

We have exploited SMD to reuse EDSO resources [23] for problem solving in GEODISE. In addition to providing ontology-enabled direct retrieval, query and search capabilities for resource consumption we have developed a semantic web based knowledge advisor integrated in various domain applications such as the

WCE and a Domain Script Editor (DSE). The knowledge advisor demonstrates one of many potential uses of SMD such as provenance and trust.

4.1 Application scenario

Figure 8 shows the walkthrough of GEODISE resource management and SMD usage in an application. In this scenario we first build a domain ontology based on function characterisation, including function classification and categorisation, function interface analysis (input/output modeling) and terminology extraction. In addition to a GEODISE domain ontology, an upper level function ontology is also provided to enable the addition of metadata to functions. These metadata could contain any arbitrary information that can help use the resources, mostly those such as authors, copyright, version, license and generic function information. Then we use the Function Annotator to annotate and publish GEODISE function descriptions into the Function/Workflow Repository. The populated repository is then ready for use for knowledge intensive applications.

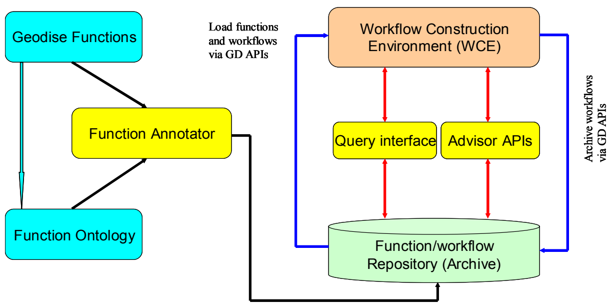

Geodise Functions
Function Annotator
Function Ontology
Load functions and workflows via GD APIs
Workflow Construction Environment (WCE)
Query interface
Advisor APIs
Function/workflow Repository (Archive)
Archive workflows via GD APIs

Figure 8. Usage scenario in GEODISE

EDSO is a process in which different computation resources are aggregated into a workflow to solve a specific engineering design problem. Problem solving amounts to constructing and executing a workflow. For this purpose GEODISE has developed a Workflow Construction Environment (WCE) (see Figure 8) to help engineers construct workflows via flexible, rapid assembly and disassembly of EDSO resources. However, building a workflow for a problem requires a lot of domain knowledge, for example, what algorithm is suitable, who has this code, where it is, how to specify the control parameters, etc. We have made use of SMD from the Function/Workflow repository to answer these questions. As can be seen in Figure 8, WCE uses high level GD API, a suite of tools for manipulating resource SMD built on top of the OWL API (http://wonderweb.man.ac.uk/owl/), to load functions and workflows from the Function/Workflow Knowledge Repository. As semantic metadata regarding these functions has been explicitly modeled and represented, engineers are able to get hold of the right functions for a specific problem using relevant query tools.

Whilst the consumption of rich SMD could have many usages regarding resource sharing and reuse, our initial attempt has produced a knowledge advisor [23] based on the interface semantics matching for functions/workflows. The integrated knowledge advisor can give context-sensitive just-in-time advice on function/workflow selection and configuration, thus facilitating workflow construction.

A workflow built in this way can inherit SMD from embedded component functions. More than this the GD API also provides mechanisms to attach overall SMD to generated workflows. Therefore, WCE will not only consume functions but also generate and archive knowledge-rich workflows into instance stores for reuse.

4.2 The knowledge advisor for workflow composition

To build a workflow for a specific problem engineers need to know what resources are required, what resources are available and what should be done next given some resources. These are not trivial problems, in particular, for new engineers. Grid applications so far have proven that to discover the right resources on the Grid is hard with regards to the heterogeneity of distributed dynamic Web/Grid resources. We have developed a knowledge advisor to support decision-making during workflow construction. The system exploits SMD to provide two levels of advice on resource discovery, selection and configuration.

Process level advice: functions can only be joined together to form a valid workflow when their interface semantics matches each other, i.e., one function's inputs/outputs are semantically compatible with another function's outputs/inputs. Based on the semantic matching of function interface the advisor can suggest all functions that fit into the workflow at a specific point during a workflow construction process. Figure 9 shows a screenshot of workflow construction in GEODISE WCE. The advisor runs in the background. When activated the advisor monitors the WCE workspace. Each time a function from the left-hand side panel is selected and dropped into the composition area, the advisor will collect the function interface and its semantics from the knowledge repository. It will then carry out semantic matching and reasoning against the underlying function repository. The advisor will return a list of semantically compatible functions as shown in the left-hand side bottom panel. Users can examine these suggested functions individually to get further information until an appropriate function is chosen.

Function level advice: with rich SMD embedded in function descriptions, users will be able to obtain configuration information by following ontological links, for example, what are the type and default values of a variable, what and where the alternative similar functions are and so on. Function level advice is provided as just-in-time hands-on tips during function selection and configuration. Figure 10 shows a screenshot of workflow construction in a Domain Script Editor (DSE). The knowledge advisor integrated in the DSE shows multiple choices for function configuration, accomplishes auto-completions and suggests alternative values to assist textual workflow construction.

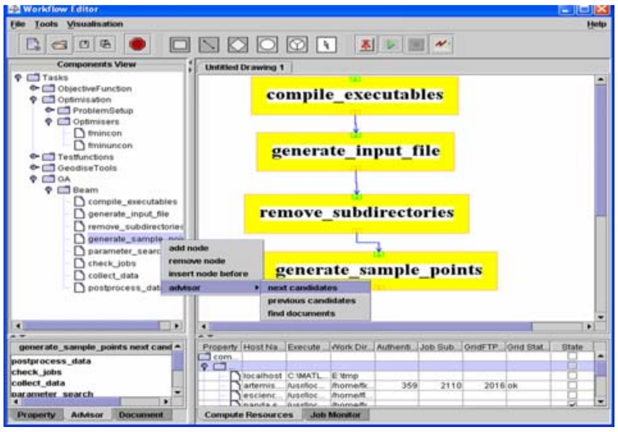
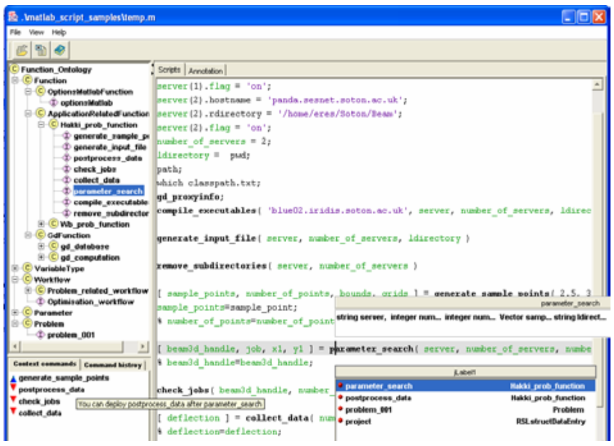


Figure 9. Knowledge advisor in the WCE


\matlab_script_samples\temp.m
Function_Ontology
Scripts Annotation
parameter_search
string server, integer num... integer num... Vector samp... string ldirect...
jLabel1
You can deploy postprocess_data after parameter_search

```
server(1).flag = 'on';
server(2).hostname = 'panda.sesnet.soton.ac.uk';
server(2).rdirectory = '/home/eres/Soton/Beam';
server(2).flag = 'on';
number_of_servers = 2;
ldirectory =  pwd;
path;
which classpath.txt;
gd_proxyinfo;
compile_executables( 'blue02.iridis.soton.ac.uk', server, number_of_servers, ldirec

generate_input_file( server, number_of_servers, ldirectory )

remove_subdirectories( server, number_of_servers )

[ sample_points, number_of_points, bounds, grids ] = generate_sample_points( 2.5, 3
sample_points=sample_point;
% number_of_points=number_of_point

[ beam3d_handle, job, x1, y1 ] = parameter_search( server, number_of_servers, numbe
% beam3d_handle=beam3d_handle;

check_jobs( beam3d_handle, number_

[ deflection ] = collect_data( num
% deflection=deflection;
```

parameter_search	Hakki_prob_function
postprocess_data	Hakki_prob_function
problem_001	Problem
project	RSLstructDataEntry

Figure 10. Knowledge advisor in the DSE

The advisor provides an effective way to reuse resources via their SMD. It can be applied to any domain as long as a semantically enriched resource repository and the underlying ontology are available. Advice on workflow construction is just one of many applications that benefit from rich SMD. Semantic metadata can be utilised for other purpose such as provenance and trust. This is future work.

5. Conclusions

In this paper we contend that managing resource SMD effectively is the key to the seamless sharing and reuse of Web/Grid resources. We have introduced a generic SMD management framework for Web/Grid resources. We have discussed issues related to semantic Web/Grid resource description, SMD storage and semantic reuse. The framework has been used for semantic resource management in GEODISE. We have developed a suite of tools, APIs and mechanisms to support the management lifecycle. These include the Function Annotator for semantic enrichment, the Function Knowledge Repository that manages large scale semantic instances and a DL-based powerful query mechanism for resource discovery.

We have generated a semantically enriched resource repository for engineering design search and optimisation in GEODISE. We have exploited the semantic metadata of resources from the repository in a knowledge advisor, which provides advice on workflow construction. Initial experience from GEODISE has proved the framework and its supportive tools are successful. In the future we plan to make use of SMD in such applications as automated service brokering, provenance and personalised service discovery. The management prototype will also be migrated to a browser-based environment, i.e. implemented as knowledge services.

6 Acknowledgements

This work is supported by the UK EPSRC GEODISE e-Science pilot (GR/R67705/01). The authors gratefully acknowledge the contributions from and discussion with EPSRC projects myGrid (GR/R67743/01) [22] and AKT (GR/N15764/01(P)).

References

1. Foster, I. and Kesselman, C. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999.
2. The Globus Monitoring and Discovering services, http://www.globus.org/mds/.
3. Foster, I., Kesselman, C., Nick, J. and Tucke S. The Physiology of the Grid. An Open Grid Service Architecture for Distributed Systems Integration, 2002, http://www.globus.org/ogsa/.
4. Web Services Inspection Language, http://www106.ibm.com/ developerworks/webservices/library/ws-wsilspec.html
5. Universal Description Discovery and Integration, http://uddi.org/ and http://uddi.org/pubs/uddi_v3.htm
6. The Advanced Knowledge Technologies (AKT) Project. http://www.aktors.org/.

7. Berners-Lee, T., Hendler, J. and Lassila, O. The Semantic Web, Scientific American, vol. 284, no. 5, May 2001, pp. 34–43.

8. Roure, D., Jennings, N. and Shadbolt, N. The Semantic Grid: A Future e-Science Infrastructure, Berman, F. et al ed. Grid Computing: Making the Global Infrastructure a Reality, John Wiley & Sons, 2003, pp. 437-470.

9. Annotation and Authoring for Semantic Web, http://annotation.semanticweb.org/org/.

10. The DAML Service Coalition: DAML-S: Web Service Description for the Semantic Web, In the 1st International Semantic Web Conference (ISWC2002), pp 348-363.

11. Benatallah, B., Hacid, M.S., Rey, C. and Toumani F. Semantic Reasoning for Web Services Discovery, In WWW2003 Workshop on e-Services and the Semantic Web, Budpast, Hungary, 2003.

12. Chen, L., Shadbolt, N.R., Goble, C., Tao, F., Cox, S.J., Puleston C., Smart P.R., Towards a knowledge-based Approach to Semantic Service Composition. Lecture Notes in Computer Science, LNCS 2870, 2003, pp 319-334.

13. Sirin, E., Hendler, J. and Parsia, B., Semi-automatic Composition of Web Services Using Semantic Descriptions, in Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003, France, 2003.

14. The GEODISE Project http://www.geodise.org/

15. Harris, S., and Gibbins, N., 3store: Efficient Bulk RDF Storage. In Proceedings 1st International Workshop on Practical and Scalable Semantic Web Systems, 2003, pp. 1-15.

16. The Instance Store, http://instancestore.man.ac.uk/

17. Horrocks, I., Sattler, U. and Tobies, S., Practical reasoning for expressive description logics. In Lecture Notes in Artificial Intelligence, No.1705, Springer-Verlag, 1999, pp.161-180.

18. Haarslev, V. and Möller, R., High Performance Reasoning with Very Large Knowledge Bases: A Practical Case Study, Seventeenth International Joint Conference on Artificial Intelligence, IJCAI-01, 2001, pp. 161-166.

19. Pound, G., Eres, H., Wason, J., Jiao, Z., Keane, A.J., and Cox, S.J. A Grid-enabled Problem Solving Environment (PSE) for Design Optimisation within Matlab. Accepted paper – International Parallel and distributed Processing Symposium IPDPS-2003, Nice, France.

20. Bechhofer, S., Horrocks, I., Goble, C. and Stevens, R., OilEd: a Reason-able Ontology Editor for the Semantic Web Lecture Notes in Artificial Intelligence, Springer-Verlag LNAI Vol. 2174, 2001, pp. 396-408.

21. Chen L., Cox S.J., Tao F., Shadbolt N.R., Puleston C., Goble C. Empowering Resource Providers to Build the Semantic Grid, Accepted for IEEE/ACM/WIC International conference on Web Intelligence WI2004.

22. The MyGrid Project http://mygrid.man.ac.uk/index.shtml

23. Tao, F, Shadbolt, N.R., Chen, L, Xu., F. and Cox, S.J. Semantic Web based Content Enrichment and Knowledge Reuse in e-Science, the 3rd International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE), Larnaca, Cyprus, 25-29 Oct, 2004