# Managing Semantic Metadata for Web/Grid Services

Chen L.[1], Shadbolt N.R.[2], Goble C.[3], Tao F.[2]
[1]School of Computing and Mathematics
University of Ulster, Newtownabbey, Co. Antrim, BT37 0QB, U.K.
{l.chen@ulster.ac.uk}
[2]School of Electronics and Computer Science
University of Southampton, Highfield, Southampton, SO17 1BJ, U.K.
{nrs, ft@ecs.soton.ac.uk}
[3]Department of Computer Science
University of Manchester, Oxford Road, Manchester, M13 9PL, U.K.
{carole@cs.man.ac.uk}

**ABSTRACT:**

Web/Grid services' metadata and semantics are becoming increasing important for service sharing and effective reuse. In this paper we present a generic framework for engineering and managing services' Semantic Metadata (SMD) with the ultimate purpose of facilitating interoperability, automation and knowledgeable reuse of services for problem solving. The framework addresses fundamental issues, approaches and tools for the whole lifecycle of SMD management, i.e. those of acquiring, modeling, representing, publishing and reusing services' SMD. It adopts ontologies and the Semantic Web technologies as the enabling technologies by which services' metadata are semantically enriched and made interoperable, understandable and accessible on the Web/Grid for both humans and machines. In particular, mechanisms are proposed to make use of service SMD for service discovery and composition. The paper also describes a service SMD management system in the context of the UK e-Science project GEODISE. A suite of tools are developed, which forms the core of the SMD management infrastructure. We demonstrate the added value of the use of SMD through the integration of SMD management with GEODISE application systems.

**KEY WORDS:**
*Metadata, Web Services, Semantic Web, Semantic Grid, knowledge management, ontology, engineering design*

# INTRODUCTION

The essence of Grid computing (Foster and Kesselman 2004) is the sharing and reuse of distributed, heterogeneous resources for coordinated problem solving. Its success relies on the effective discovery, seamless aggregation and effective use of the "right" services for the "right" problem. Metadata is data that provide extra information about other data. For example, a photo can be described using the following metadata: <dateTaken> 12/09/2003 </dateTaken>, <placeTaken> seminar room </placeTaken> and <whatAbout> GEODISE project meeting </whatAbout>. Metadata is becoming increasingly critical in Web/Grid computing because human users as well as software agents increasingly rely on metadata for service discovery, reuse and expertise sharing.

Metadata exist at all levels of the Grid, ranging from low-level repositories of resource handles to upper-level application-related services. At the time of writing, the metadata of low-level hardware-related Grid services is stored and managed by core Grid services such as Globus MDS (www.globus.org/mds) and RGMA (www.r-gma.org). In the Open Grid Service

Architecture (OGSA) (Foster, Kesselman et al. 2002) application-level resources are wrapped as Web or Grid Services, and services' metadata are associated with Web/Grid services, which are described in WSDL files (www.w3.org/TR/wsdl12), and published and stored in UDDI repositories (www.uddi.org).

The way that current service-oriented infrastructure handles and manages services' metadata is not adequate and effective for metadata to help services discovery and knowledge sharing. First, there is no enough metadata about Web/Grid services. Services, in particular, legacy resources, are developed by service providers for their own use, without realizing the role and importance of metadata this naturally leads to the lack of descriptive information for services. Second, metadata are unstructured. Web/Grid services are diverse; the types of metadata required for describing services in e-Science (Hey and Trefethen 2003) vary greatly between individuals, organizations, and scientific communities. The use of different terminologies and the adoption of various metadata models such as using comments or annotations as metadata are inevitable. Unsurprisingly, this causes the problem of mutual understanding and service interoperability. Third, metadata lack semantics. XML (www.w3.org/XML) based metadata modeling and representation as in WSDL and UDDI are incapable of capturing genuine semantics, relationships or constraints. There are no problems for humans to understand XML-based metadata as described in the above photo example because we know the meaning of these English words. The question is: "can machines understand and consume them?" so that they can perform automated and automatic processing with regards to the use of Web/Grid services. Clearly without further assumptions, the answer will be no. Fourth, there are no dedicated metadata storage and associated query and reasoning facilities. UDDI is not supposed to deal with large amount of metadata. While it is possible to incorporate rich metadata into UDDI repositories, UDDI itself does not provide scalable storage mechanisms and rich capabilities for manipulating metadata, such as query and reasoning against metadata.

The Semantic Web (Berners-Lee, Hendler et al.2001) and Semantic Grid (De Roure, Jennings et al. 2003) (Zhuge 2005) are extensions of the current Web/Grid in which information and services are given well-defined meaning, better enabling computers and people to work in cooperation. We believe that the first step towards the Semantic Web/Grid is to make the Web/Grid full of rich SMD, i.e. metadata with semantics. To achieve this objective, we argue that an integrated framework for SMD modelling and management is required so that service's metadata are flexible and extensible, and metadata generated in such a way have explicit, conceptually consistent meaning. This framework ought to exploit knowledge engineering techniques in advanced knowledge technologies (Zhuge 2005) (Goble, De Roure et al. 2004) (www.aktors.org/akt), the emerging infrastructure in the Semantic Web and Web/Grid services technologies (www.w3.org/2002/ws/) (www.globus.org/wsrf) communities in order to work with heterogeneous distributed services across dynamic virtual organizations.

This paper proposes a generic framework for SMD engineering and management, which can be applied to both Web Services and Grid services. Our contributions are three folds. Firstly the proposed framework provides a systematic, coordinated approach to SMD management, offering a new and deep understanding of SMD management as a crucial means for service-oriented computing. Secondly various methods and mechanisms are proposed to addresses the complete life cycle of SMD management based on the state of the art of the research on the Semantic Web, Web Services and the Grid. Thirdly a reference implementation for the framework is developed and deployed in the context of Grid Enabled Optimisation and Design Search in Engineering (GEODISE) (www.geodise.org). The implementation not only verifies our approach but also provides an infrastructure for applying this approach to real world applications. We demonstrate the benefits of using SMD for problem solving.

The paper first introduces the framework for SMD management, which include detailed analyses and discussions for each constituent component regarding its functionality and

realization approaches. Second we describe the design and implementation rationale of the SMD management framework in the context of GEODISE. Then we present the deployment of the SMD management system and demonstrate its usage and benefits in service composition. Experience and lessons learnt from the research are also discussed. Following this we review related work and point out our contributions. Finally we conclude the paper by discussing future extensions and improvements.

# A SYSTEMATIC FRAMEWORK FOR ENGINEERING AND MANAGING SERVICE'S SMD

Semantic metadata refers to the metadata that are formally modeled based on their context, thus giving them meaning. Service SMD is actually a type of knowledge about the service's general characteristics, interfaces and execution details. By this view, we argue that SMD management should have its own lifecycle, namely those of acquiring, modeling, representing, publishing and reusing a service's SMD. While traditional knowledge management technologies (Schreiber, Akkermans et al. 1999), which provide methods, templates, protocols and tools to support the lifecycle of knowledge, are available, they are not designed for distributed knowledge management. For example, most of knowledge models, templates and representations for standalone applications cannot be recognized, shared and reused by Web/Grid users and applications.

Inspired by the latest research results on ontologies and the Semantic Web, we conceive an ontology-centric approach to SMD management. The key features of the approach are as follows: Firstly, ontologies are used for metadata and context modeling, thus help towards interoperability and machine understandability. Secondly, knowledge acquisition, i.e. service metadata collection and semantics tagging, is carried out semi-automatically through a formal knowledge binding process – also known as semantic annotation. Thirdly, web ontology languages are used for SMD knowledge representation, thus enabling knowledge sharing and effective reuse.

Figure 1 shows the SMD management framework, which specifies key components and their interactions. In this framework we abstract away low-level Web/Grid compute fabrics and focus on application-level services and their metadata management. We take a broad view with regards to Web/Grid service type and representations, i.e. services could be Web/Grid services and/or any other forms of computational algorithms and functions. As can be seen in the figure, the framework consists of DomainLayer, ApplicationLayer and WrapperLayer. The DomainLayer contains domain-specific services, knowledge and metadata. The ApplicationLayer refers to various systems that make use of domain services and knowledge for problem solving. In standalone computing environment, the ApplicationLayer will be directly on

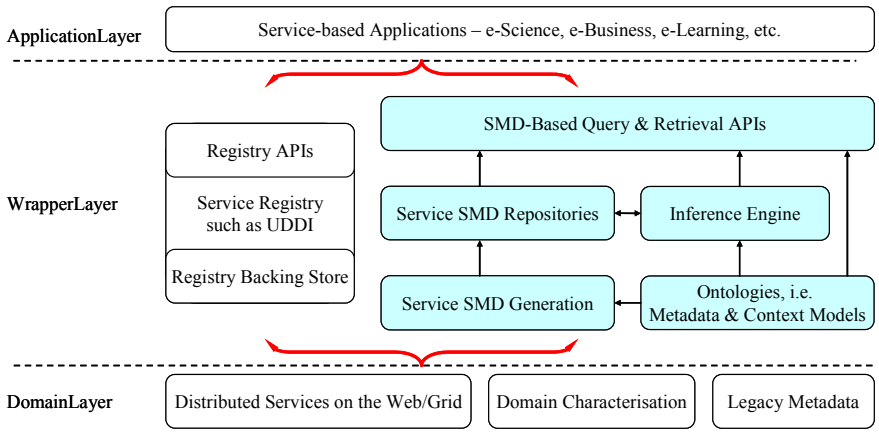


Figure 1: The Generic SMD Management Framework

top of DomainLayer and application systems will only use local domain services for accomplishing tasks. To enable distributed applications to access and use distributed services on the Grid the WrapperLayer is needed to facilitate service wrapping, publishing, discovery and reuse.

The left module of the WrapperLayer presents the current mechanisms of service discovery, access and sharing for SOA and OGSA based applications. The scenario is that service providers publish services into public registries such as a service registry and service indexes. Service consumers discover required services from public registries in terms of key words and/or functionality signature. The lack of rich metadata, in particular their explicit semantics, makes current Web/Grid applications usually involve huge amount of front end preparations such as human interpretation and manual configuration.

The right module of the WrapperLayer shows the anatomy of SMD management system. It is intended to replace present service registry with SMD-based service management system with its ultimate goal of creating a SMD rich Web/Grid environment. The added values of SMD management are the promotion of service interoperability, machine understandability and automation. Detailed descriptions about functionality, methods and techniques for each component, and their limitations and advantages are described below.

## Ontologies: Metadata and Context Modeling

This component intends to capture all metadata of Web/Grid services and the concepts related to domain in which these services operate. It further models these metadata, concepts and their relations in a structure using commonly agreed terms. The purpose is to abstract the ontological entities of metadata and put them in context, thus giving them meaning.

Metadata need to be interpreted with respect to a context, i.e. the domain the metadata are about. The interpretation is actually the assignment of meaning to metadata symbols in terms of domain concepts and/or theories. As the Web/Grid is an open world environment encompassing a wide variety of fields, communities and areas, a service's metadata may be given different meaning dependent on the users' understanding and background knowledge about the service. Such multiple and even ambiguous interpretations will also prevent machines and software agents from automatically processing metadata because they do not know in which domain the metadata should be interpreted. Therefore, a formal context modeling is required.

In knowledge representation, context modeling amounts to creating an abstract, simplified view of the world that we wish to represent for some purposes. Given that context is used for metadata interpretation in our case, metadata modeling will go hand in hand with context modeling.

Our framework uses ontologies to perform metadata and context modeling in which entities such as services will be conceptualized as ontological concepts and an entity's metadata will be conceptualized as its properties. For instance, in the above photo example, the photo will be modeled as an ontological concept and metadata such as dateTaken, placeTaken and whatAbout will be modeled as the properties of the photo concept. Context modeling will conceptualise all other entities related to the concerned entity and establish relations among them via concepts' properties. Overall context modeling will create a self-contained ontology in which metadata can be interpreted unambiguously by both humans and machines.

Ontology-based metadata and context modeling provides a common communication language for Web/Grid service providers and consumers. It is normal that service providers and/or consumers in different virtual organizations refer to same things with different terms (and/or different things with same terms). For instance, a photo may be described using the following set of metadata: <dateTaken>9:00am, 12/09/2003</dateTaken>, <venue>seminar room</venue> and <content> GEODISE project meeting</content>. These metadata are different from those used in

the previous example. This means that a person who adopts this set of metadata for photo publishing will be not able to discover photos published in terms of the previous metadata and vice versa. The use of ontologies will bridge up the gap by providing a commonly agreed terms, thus enhancing the interoperability. It also allows for flexibility and adaptation to accommodate diverse metadata and future changes within the field.

## Service SMD Generation

Ontology-based metadata models are conceptual templates, i.e. knowledge-preserving structures. To generate SMD, it is necessary to bind metadata models with the concrete information of the concerned services. This component consists of two tasks – metadata collection and metadata instantiation with metadata models (ontologies). In view of the nature of heterogeneity, distribution and the dynamics of Web/Grid environment, SMD generation pose a great challenge for SMD management.

Two approaches are identified for capturing services' metadata: the human-centered approach and information extraction based approach. In the first approach, a person (either a service provider or a domain experts or a knowledge engineer) analyzes service domain, obtains all metadata values and prepares them in accordance with the metadata model. This approach requires that the person should have domain background knowledge. The latter approach is to extract metadata values using information extraction techniques. It tries to acquire metadata automatically by parsing and recognizing designated entities and their values. The problems with this approach are that different service providers may use different terminology for their services. An information extraction algorithm that works for one domain may not work for others. Furthermore, some services, in particular those legacy services, may not have enough information.

SMD is generated through metadata instantiation and semantic enrichment. Metadata instantiation is to assign values to metadata. Semantic enrichment is to establish links between the services (concepts), metadata (properties) and metadata assignments (fillers). By following ontological links metadata and their assignment can be explicitly defined in terms of ontological concepts, properties, values and relations. These links allow both humans and machines to track down the exact meaning of metadata and their assignments based on the ontology – context model. This guarantees metadata can be interpreted unambiguously.

There are three ways of generating a service's SMD, i.e. manually, automatically or semi-automatically. The manual approach involves manual metadata collection and binding. It is tedious and time consuming. This has been the bottleneck that hinders the development of the Semantic Web. The automatic approach is to use information extraction and natural language processing techniques to perform metadata collection and binding by software agents. This really depends on a service's domain and its characteristics. It is highly likely to carry out automatic metadata collection and binding with services that have well-structured data and/or well-represented interfaces. In reality it proves hard to generate SMD automatically because the lack of standards and conventions for the terminology and representation of distributed Web/Grid services has made automatic IE very hard. Therefore, in general a semi-automatic approach involving both IE tools and human interactions is more realistic. In such approach domain-oriented dedicated IE tools are developed to perform information extraction to obtain as many metadata values as possible. The SMD creator will manually collect and bind the missing metadata.

## Service SMD Repositories

Service SMD repositories component is responsible for service SMD representation and storage, which are described below.

**Service SMD representation**

SMD representation needs to fulfill several requirements. First it should have appropriate expressive capabilities, thus being able to model and convey all explicit meaning of metadata without any ambiguity and fidelity loss. Second it should be easily distributed and accessed on the Web/Grid so that as many Web/Grid users as possible can get hold on it. Third SMD representation should allow for high degree interoperability and machine understandability in order to facilitate SMD processing and semantic consumption for end users' applications.

Many languages have been designed to express the ontology and semantic information. Among them, the most recent is the Web Ontology Language (OWL) (www.w3.org/2004/OWL), which has evolved from RDF (www.w3.org/RDF/) and DAML+OIL to provide more expressive power. RDF is a graph model (or sets of triple statements) which is designed for describing and searching services on the Web. DAML+OIL (www.daml.org) is a schema language that adds constraints on properties to assist machine reasoning. For example when "daml:TransitiveProperty" is added as a constraint on the property "P1:older_than" of a RDF model, if we have A1:P1:A2 and A2:P1:A3, then A1:P1:A3 can be inferred. This is useful for reasoning and inferring new knowledge that has not been directly stated. DAML+OIL also uses subProperty to describe relationship at different granularities. OWL is built on DAML+OIL with additional constraints such as "sameAs", "cardinality", etc. for more expressive power. Both DAML+OIL and OWL are based on the knowledge representation formalism of Description Logic (DL), which gives OWL a solid foundation on which semantics can be explicitly expressed and reasoned.

RDF is well established in the Semantic Web communities as a knowledge representation language. There exist various open source tools, APIs and diversity of RDF repositories for RDF applications. OWL has recently become W3C standard. It also provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users. With many open source tools and APIs being developed, OWL is getting its currency.

Which language to use for SMD representation is actually a question of choice, depending on application characteristics, users' preferences and the way SMD is used. For applications that involve large amount of ontological concepts, thus requiring consistency check and classification OWL might be a better choice. OWL is also appropriate for applications that need description-logic based reasoning.

**Service SMD Storage**

So far we have not defined what exactly a service's SMD is in terms of formal metadata models and representation. In ontology terminology, the SMD of a service is the instance of the ontological concept of the service. Alternatively we can say a service's SMD is the semantic description of the service using metadata and context models. Concretely a service's SMD is a number of instantiated schema interconnected via ontological links with each schema filled of concrete values. SMD could be represented using different knowledge representation formalisms such as RDF and OWL.

There are three different mechanisms to store a service's SMD. Firstly it can be embedded into the original service as a set of descriptions. Some annotation tools such as the OntoMat-annotizer (annotation.semanticweb.org/ontomat/index.html) in the Semantic Web community use this mechanism to attach semantic descriptions to web pages. Secondly, SMD can be saved in a separate storage in the same location as the service is stored. A local reference link will establish the relationship between a service and its SMD. Thirdly SMD can be archived in distributed

knowledge repositories separate from services. Globus MDS and Web Service registry such as UDDI have adopted this approach to archive metadata.

As we can not attach text into non-text files or documents, the first mechanism will not apply to services that are modeled and delivered as binary such as executables. Its another shortcoming is that the overall size of service plus its SMD may become unmanageable in a service registry. It could also incur extra processing costs because both services and their SMD will be accessed and loaded no matter which one is actually required. The second mechanism is straightforward. However, as there are service registries for services and no registries for services' SMD, to access a service's SMD needs to first find the service from a service registry. This will not serve the purpose of SMD as a mean for service discovery as discussed previously. In the context of Web/Grid computing it is supposed that services are owned by and geographically located in dynamic virtual organisations. These services should be published with explicit expressive descriptions exposing as much information as possible, so that they can be discovered, shared and reused. From this perspective the third approach, i.e. archiving SMD in a (or several distributed) central knowledge repository, seems ideal for realizing Web/Grid vision via SMD management. Such knowledge repository can also serve as a registry service similar to the UDDI registry but with rich SMD.

A critical issue in creating SMD repositories operating on the Web/Grid is scalability with regards to the storage, retrieval and reasoning of SMD entities. Real world Web/Grid applications usually involve dozens of ontologies, which may consist of thousands of concepts and hundreds of properties, and large numbers of semantic instances. For example, in AKT project (www.aktors.org), the hyphen.info (www.hyphen.info) dataset consists of around 5 million RDF triples when serialised, and is expected to grow to several tens of millions if fully populated. The base design scale of the knowledge repository is set to be the ability to handle at least 20 million triples and 5000 classes and properties. In GONG project (gong.man.ac.uk/) researchers have built a semantic knowledge repository that consists of a gene ontology of 50,000 concepts and up to 650,000 individuals. While existing techniques for terminological reasoning (i.e., reasoning about the concepts in an ontology) seem able to cope with real world ontologies (Horrocks, Sattler et al. 1999) (Haarslev and Möller 2001), it is not clear if existing techniques for assertion reasoning (i.e., reasoning about the individuals in an ontology) will be able to cope with realistic sets of SMD. This difficulty arises not so much from the computational complexity of assertion reasoning, but from the fact that the number of individuals (e.g., SMD) might be extremely large.

Technologies for scalability problems are closely related to SMD representations and the inference mechanisms that can be applied to the representations. Currently there are two mainstream knowledge base technologies for SMD storage, retrieval and reasoning, which are mainly categorized in terms of SMD representation. The first one is based on the RDF formalism. Systems using this technology include Sesame (Broekstra, Kampman et al. 2002) and 3Store (Harris and Gibbins 2003). The second one focuses on DL-based descriptions described by OWL. Such systems include RACER (Haarslev and Möller 2003) and Instance Store (*IS*) (Horrocks, Li et al. 2004). The common approach of these systems is to use database technology for SMD instance indexing and search optimization, and semantic inference mechanisms for the classification of ontological concepts. By replacing reasoning with SMD instances with reasoning with concepts and optimized database search, the retrieval and query performance can be significantly improved.

While further extensions and formal experiments and evaluations are needed for SMD knowledge base technologies, nevertheless these systems, in particular, the 3Store and Instance Store, provide a starting point for SMD management. Once again the development and/or the selection of SMD repository technology would depend on the nature of the application and the characteristics of the KB.

## Inference Engine

Inference Engine provides reasoning capabilities for service SMD management system. It has two main usages: First an inference engine can be used to help construct a large ontology by performing such actions as subsumption, classification, concept consistency check, etc.; Second, to discover a specific service in terms of user query criteria an inference engine is needed to reason against the SMD repositories.

There are different ontological reasoning engines. For instance, the FaCT reasoner (Horrocks, Sattler et al. 1999) can perform terminological reasoning; the RACER reasoner can perform instance reasoning. Requirements for a reasoning engine and its inference capability are closely related to the SMD representation and storage as discussed in **Service SMD Storage**, it is also pertinent to the SMD use in applications.

### SMD-based Query and Retrieval APIs

Once service SMD repositories are populated with SMD, Web/Grid service consumers can make use of the semantic information for many purposes. This component is responsible for providing semantic information consumption mechanisms and tools to facilitate the use of service SMD. General speaking, SMD can used in the following ways: Firstly consumers can navigate services in the repository in terms of SMD. Services and metadata are classified into different categories when they are formally modeled using ontologies. By referencing the associated ontology users can obtain all services under a specific service category (a concept and/or a property) and their SMD. These services can be presented in a hierarchical structure that shows their inter-relations and also facilitates selection.

Secondly service consumers can exploit SMD for service discovery. The above functionality, i.e. service browsing, is desirable but may not be practically realistic when the size of SMD instances grows to thousands or millions. Semantics-based search is different from traditional keyword-based search mechanism in that service matching is based on meaning rather than signatures. For example, by specifying a set of metadata and value pairs, DL-based reasoner can discover a number of services that have these metadata. This not only increases the accuracy of service discovery, enhances interoperability, but most importantly, enables automated machine processing.

Depending on the richness of knowledge captured through metadata modelling, SMD can be exploited to different extent for application specific purposes. An example is semantics-based recommender systems for service composition and aggregation. Service can only be joined together to form a valid workflow when their interface semantics matches each other, i.e., one service's inputs/outputs are semantically compatible with another service's outputs/inputs. Based on the semantic matching of service interface a recommender system can suggest all services that fit into the workflow at a specific point during a workflow construction process. The recommendation can also be given in service level for service configuration such as what are the types and default values of a variable, what and where the alternative similar functions are and so on.

The extent to which the SMD can be used for Web/Grid applications is dependant on how much SMD is available on the Web/Grid and how much knowledge the SMD holds. The more knowledge in the SMD, there will be the more SMD usage in Web/Grid applications. The more SMD are available on the Web/Grid, the closer it is for the Web/Grid to move to the so-called Semantic Web/Grid. To facilitate Web/Grid service consumers to access and retrieve services in terms of SMD, APIs and tools are needed. This will be discussed in details in the context of GEODISE.

# SMD MANAGEMENT IN GEODISE

Grid enabled optimisation and design search in engineering (GEODISE) 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, geometry modeling and meshing packages, analysis codes and distributed computing and data services on the Grid. Extensive analysis of user requirements and application scenarios has revealed that the key issues to achieving GEODISE objectives are (1) how to add rich metadata to GEODISE services, (2) how to semantically enrich them, and (3) how to allow sophisticated reasoning and query capabilities over them. To this end, we have adopted our framework for SMD management in GEODISE.

## GEODISE SMD Management System

EDSO has been practiced for decades, and it has accumulated huge amounts of expertise and algorithms in research institutions and commercial enterprises in various formats. Based on survey results of current EDSO practices, GEODISE has decided to use Matlab, a widely adopted engineering package in academia and industry, as its problem solving environment (Eres, Pound et al. 2005). The main resource format in Matlab is function scripts - a type of high-level computation programs that can accomplish various EDSO tasks by execution in Matlab environment. This means that application level services in GEODISE are Matlab functions.

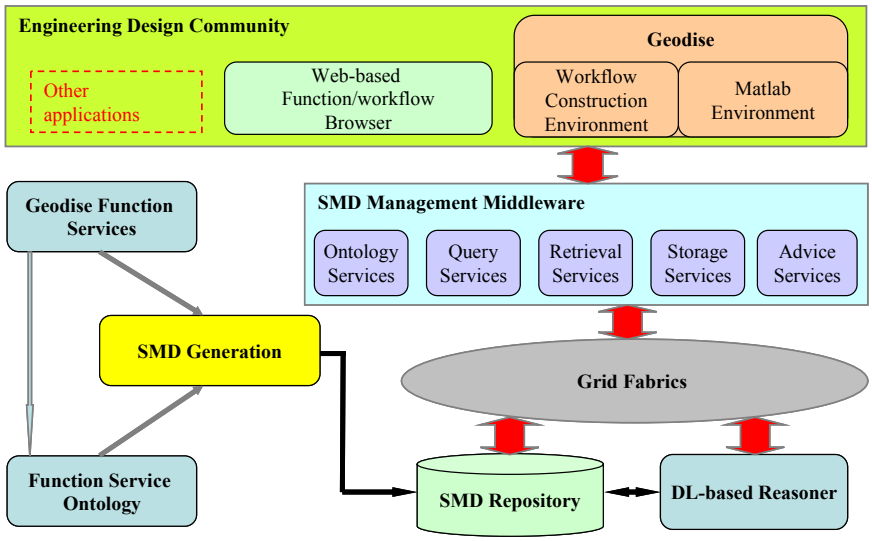


Figure 2: Function Service SMD Management System in GEODISE

Figure 2 shows the function SMD management system in GEODISE. The system implements the framework for SMD management and focuses on a domain specific type of service, i.e. Matlab functions. A key feature of the system is that it adopts a service-oriented approach to distributed SMD management. It implements all activities related to SMD consumption and supply as knowledge Web Services. This approach has two main advantages. Firstly it fits naturally into the service-based Grid infrastructure, thus easy and straightforward to be integrated into present Grid infrastructure. Secondly it provides a common mechanism for Grid users to publish, retrieve and search services' SMD, thus facilitating Grid service sharing and reuse.

In the following, we briefly describe the design rationale and implementation details of each component of GEODISE SMD management system; we also discuss the lessons learnt from the work.

## GEODISE Function Ontology

The design rationale of GEODISE function ontology is two folds. First it intends to provide formal metadata models so that functions can be described using these commonly agreed models and terms. Second it aims to provide formal context models so that metadata can be unambiguously interpreted.

Ontology development is a knowledge modeling process. In accordance with the CommonKADS methodology, a number of activities should be done before knowledge models can be built. Following these guidelines, in GEODISE we first carry out domain analysis to identify application scenarios and knowledge intensive points. Second we conduct initial knowledge elicitation and technology survey. Based on results from these activities, we finalise the areas we shall provide knowledge support, identify knowledge application scenarios and propose initial knowledge system components. Then we concentrate on knowledge acquisition and modeling, i.e. ontology development. More details can be found in (Chen, Cox et al. 2002).

We build a domain ontology, which covers the fundamental concepts and properties in EDSO, including variable type, unit types, parameter meaning and algorithm classification, and a function ontology, which is based on function classification and categorisation, function interface analysis (input/output modeling) and terminology extraction. The function ontology is based on the OWL-S (www.daml.org/services/) ontology in which we use function profiles to describe function metadata. Semantic descriptions are generated when linking service metadata and interface (inputs/outputs) with underlying EDSO domain concepts.

Figure 3 shows GEODISE function ontology. The left hand panel displays the hierarchical structure of the EDSO function-related concepts. The right hand panel lists metadata for Matlab functions.

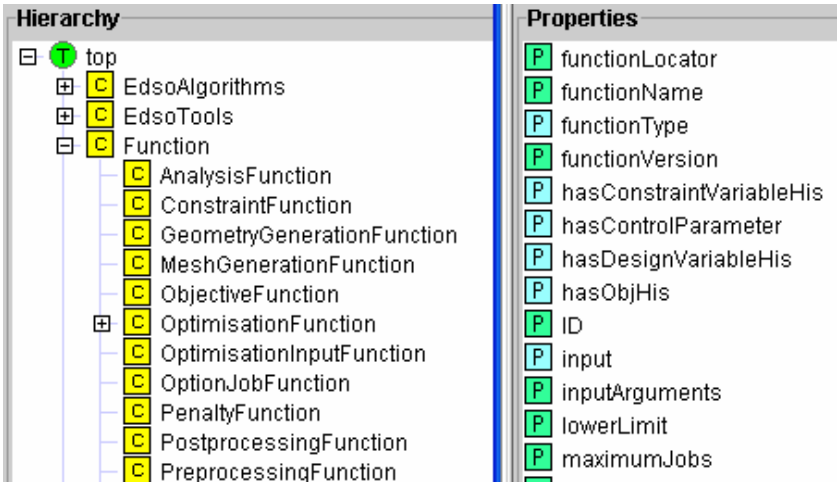


Figure 3: GEODISE Function Service Ontology

It is worth pointing out that metadata modeling is not a separate standalone activity but closely related to overall application objectives, application scenarios and potential knowledge usage. Initial domain analysis, scenario development, identification of appropriate knowledge sources and technology bottlenecks are critical for work at later stages. The lesson we get is to get domain experts, application developers, knowledge engineers, knowledge system developers and end users on board as earlier as possible. Comments, opinions and feedback from different stakeholders are valuable to come up a feasible working plan and drive the project into the right directions.

## Generating Function Services' SMD

In GEODISE, services are Matlab functions, which are written following some conventions. For example, a Matlab function script always starts with the output arguments followed by function signature and then the input arguments; there are also layout rules in the structure of a function script body. Therefore, information extraction and pattern recognition techniques can be applied to Matlab function scripts for automatic metadata extraction. While it is impossible to obtain all information required because not all function providers follow the conventions, nevertheless the ability to be able to extract information from functions will significantly reduce the workload of SMD generation.

We have developed a tool, called Function Annotator (Chen, Cox et al. 2004 ), to accomplish the job of generating functions' SMD and publishing them into a SMD repository. Function Annotator, (see Figure 4) 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), which presents the terms, relations and hierarchy of the function ontology, and a function hierarchy (Panel 2), which displays semantically enriched functions and their SMD.

Function Browser in the right-hand column is used to load Matlab functions for SMD generation. Its top window (Panel 5) contains the extracted metadata such as functions' inputs and outputs, copyright, authors and summary. Its bottom window (Panel 6) displays the source code of a Matlab function that gives users more flexibility for annotation. The Annotation Palette in the middle column performs semantic enrichment. The top window (Panel 3), the Function Profile, generates SMD for such metadata 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. The bottom window (Panel 4), the Function Model, generates SMD for such metadata as how a function works and how it can be invoked. This includes input/output arguments, location and expression signatures.

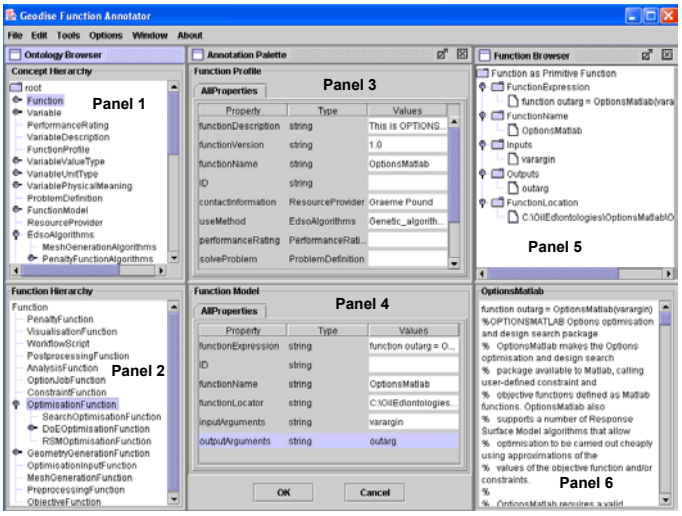


Figure 4: Function Service SMD Annotator Interface

SMD generation is actually a knowledge population process. This can be done by either knowledge engineers or domain experts or service providers. For knowledge engineers to do the job the advantage is that they have dedicated expertise on knowledge engineering and modeling and also skills for knowledge engineering tools; the disadvantage is that they do not have domain knowledge, a knowledge source must be available for knowledge input. The advantage for service provides or domain experts to do the job is that they know what metadata are important and should be captured and modeled, but they are usually not capable of using complex knowledge engineering tools and also no ideas how to organize knowledge in an appropriate way.

Given the nature of Grid computing and also from our experience in GEODISE, it is more realistic for service providers or domain experts to generate SMD by themselves. The main reasons are that: (1) Domain experts as in GEODISE are keen to be able to capture knowledge and publish it. They want the flexibility and capability. (2) It is not feasible for knowledge engineers to be available each time a service becomes available and its SMD need to be generated.

We discover that the key for service providers to generate SMD is easy-to-use tools. We recognize there is a shift of role for knowledge engineers, i.e., knowledge engineers should not

focus on knowledge population by themselves but provide tools for domain experts. Function Annotator is designed following this rationale to enable service provider generate SMD. For example, most forms in Function Annotator are generated automatically from the underlying ontology; fields are filled automatically wherever possible; suggestion and recommendation are popup wherever appropriate. Function Annotator has been used in GEODISE by service providers for SMD generation.

## SMD Representation and Repository

While various technologies are available for SMD representation and storage, EDSO service characteristics and the usage of GEODISE SMD determine that OWL is the most suitable for representing GEODISE ontologies and functions' SMD. EDSO services usually embody deep knowledge about their usage and performance. For example, there are dozens of different optimization methods, each of which is geared to solving a specific type of engineering problem. Even with a single method, different configurations of control parameters may produce very different results. To model such complex knowledge requires OWL's expressive capability. The subtlety of knowledge of EDSO services also means service discovery is dependent on detailed search criterion. Description based search mechanism is the best approach to dealing with this requirement. Services can be discovered in terms of the descriptions; the more detailed descriptions, the more accurate service will be found.

In GEODISE we have adopted the instance store technology to create a functions' SMD repository. The SMD repository uses a relational database as its permanent storage media and the DL-based reasoner RACER (www.sts.tu-harburg.de/~r.f.moeller/racer) to support reasoning. A function's SMD, i.e. an ontological concept instance, is stored in the database together with information inferred using the RACER 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 functions' SMD 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 (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 versa.

```
<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>
```

Figure 5: A Fragment of DIG Instance Representation

## SMD Management Middleware

In GEODISE we have developed a number of generic knowledge services, including front-end GUIs, to support the SMD management. While new services can be added any time in terms of application requirements, a set of core services have been identified indispensable, which are described below.

**Ontology services** enable users to access and share ontologies available on the Grid. They provide access to concepts, their properties and relationships in an underlying ontology data model. Services include retrieving definitional information, navigating concept hierarchies and retrieving lexical information.

**Storage services** are intended to publish and store a service's SMD in SMD repository for sharing and reuse on the Grid. Traditionally, knowledge bases have been small, standalone, locally stored, and consumed by the developer. Grid-oriented SMD knowledge repository may be federated, distributed, developed and consumed by multiple Grid users. Storage services enable Grid service providers to publish their services' SMD and subsequently facilitate service discovery and reuse.

**Retrieval services** allow Grid service consumers to retrieve all services and their SMD in view of the requirements of applications. A front end function browser, as shown in Figure 6 uses the retrieval services to access all services in function's SMD repository. The left hand panel displays a function hierarchy. The right hand panel displays the SMD of the selected service.

**Query services** provide APIs for two tasks. The first is the construction of query expressions and the transformation of the expressions to DL formats. The second task is to discover required services in terms of query criteria. Service discovery is carried out through SMD matching that is performed by the RACER reasoner.

**Advice services** make deep use of SMD to make recommendation on service selection, composition and configuration, which are discussed in details in the following.

These core services form the main fabrics of SMD management system, which should also be viewed as a part of the Semantic Grid infrastructure. Except the storage services all services have been implemented in the form of both traditional APIs and Web Service interfaces, thus enabling local and remote use (via WSDL). The service-oriented implementation of SMD management functionalities enhances the accessibility of the SMD repository, and further the sharing and reuse of services in the repository. By combining different services, Grid applications can realize a diversity of knowledge-based functionalities with regards to user requirements.

Apart from the function browser (see Figure 6), a front end query GUI is also developed to assist semantics-based service discovery, see Figure 7. The GUI makes use of ontology services and query services. The query criteria, query-building forms and the fillers' values of query criteria are all generated automatically from the ontology, realized via the ontology services. The GUI consists of a dropdown list and three panels. The dropdown list contains all query criteria - the metadata types of services. All selected query criteria will be displayed in the left-hand side panel for building up an overall query expression. The right-hand side panel has three purposes. Firstly it is used to show query results. Secondly it displays all available ontological concepts and/or instances for a selected metadata type. Users can assign a concept or instance to a query criterion. For example, the right-hand side panel of Figure 7 shows the hierarchy of optimization algorithms. Thirdly for the criteria of a primitive data type this panel is a textual editor. Users can directly input the value for the selected criterion. The middle panel

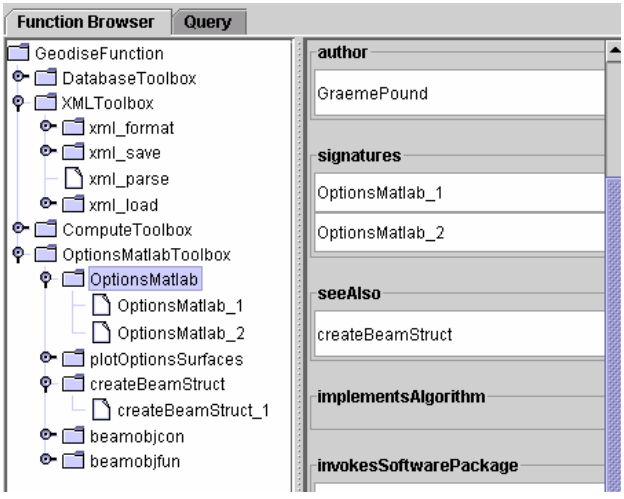


Figure 6: Function Service Browser GUI

contains a number of control buttons to facilitate query expression construction.

When the "Run Query" button is clicked all query criteria's values will be collected and a query expression can be built up. This expression will be initially represented as XML in client side, passed onto server side via HTTP and then transformed into DL formats. The underlying DL reasoning engines will reason against the SMD repository to obtain a set of entities matching all of the specified criteria. The results are displayed in the right-hand side panel.

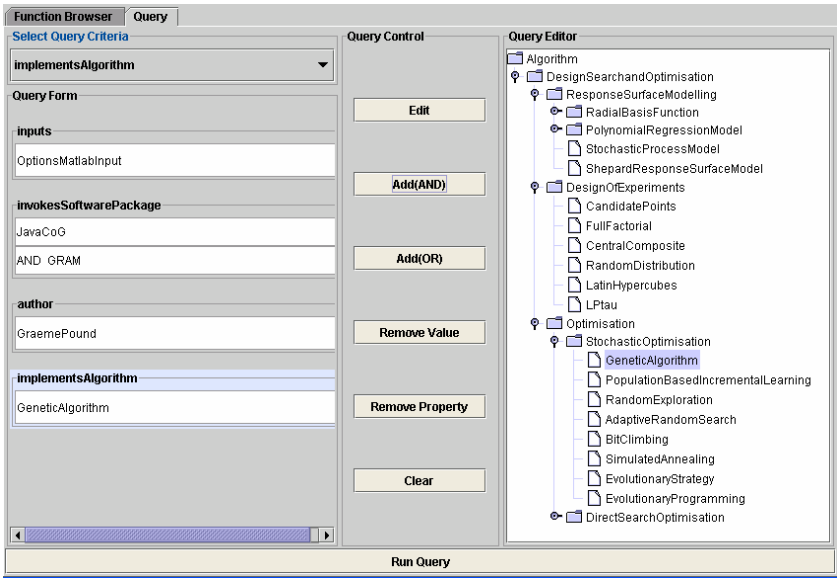


Figure 7: Function Service Query GUI

One observation from GEODISE is that DL-based reasoning over instances is quite slow comparing to both RDF-based instance reasoning and traditional database systems. This suggests that if an application does not require OWL's expressive capability and DL-based reasoning, other ontology languages such as RDFs can be adopted. The choice of knowledge representation formalisms could also incur the use of different set of tools, including ontology editor, APIs and query languages. The lesson is that technologies must be selected based on application nature and requirements.

# Using SMD for Problem Solving in GEODISE

This section describes the deployment of GEODISE function's SMD management system and an example usage of SMD for assisting engineers for problem solving.

### SMD System Deployment

Figure 8 shows the deployment of GEODISE SMD management system. The Server Side hosts function ontologies, function's SMD repository and a DL-based reasoning engine; each of them can run in a distributed separate server as long as the server is on the Grid. The Client Side could be any domain specific applications that solve problems by consuming SMD. In GEODISE these include a script-based Matlab execution environment and a GEODISE specific workflow construction environment. Client-side applications access and manipulate function's SMD through SMD management middleware, i.e., the low-level core SMD management services and high-level domain tailored tools and/or front-end GUIs.

As GEODISE uses Matlab as its execution environment, we provide a particular type of high-level middleware called SMD management toolbox. The toolbox consists of a number of Matlab functions that enable Matlab users to invoke SMD services to access a SMD repository, retrieve functions and query SMD from Matlab environment.

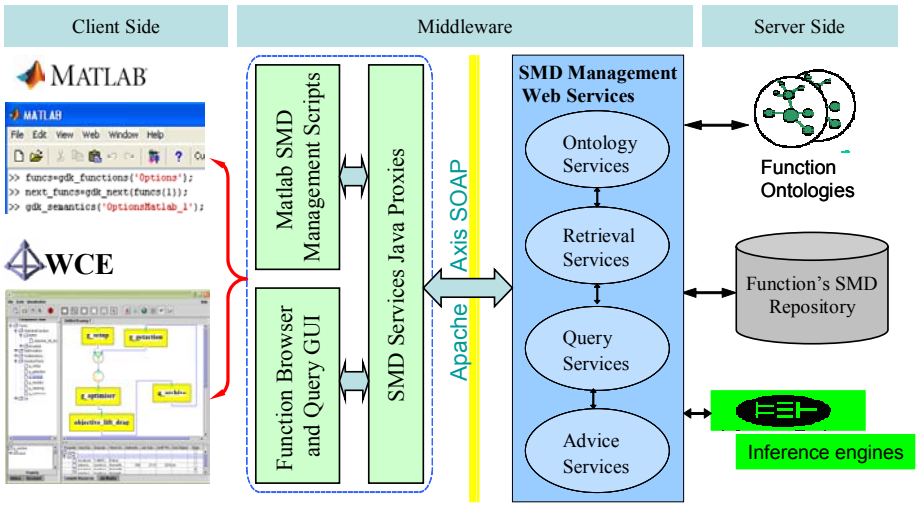


Figure 8: SMD Management System Deployment in GEODISE

EDSO is the process whereby engineering modeling and analysis are exploited to yield improved designs. An EDSO process usually comprises many different tasks. Consider the design optimization of a typical aero-engine or wing. It is necessary (1) to specify the wing geometry in a parametric form which specifies the permitted operations and constraints for the optimisation process, (2) to generate a mesh for the problem, (3) decide which analysis code to use and carry out the analysis, (4) decide the optimisation schedule, and finally (5) execute the optimisation run coupled to the analysis code. Apparently a problem solving process in EDSO is a process of constructing and executing a workflow.

To build a workflow for a specific problem, engineers need to know what services are required, what should be done next and how to configure an algorithm's control parameters, etc. These are not trivial problems, in particular, for new engineers. Based on the SMD management infrastructure, a knowledge-based recommender system is developed to support decision-making during workflow construction in GEODISE. The recommender system can give context-sensitive just-in-time advice on service discovery, selection, composition 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 recommender system can suggest all functions that fit into the workflow at a specific point during a workflow construction process.

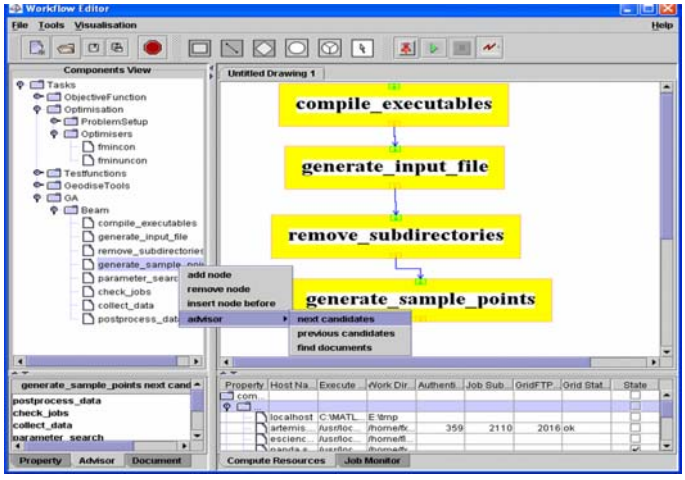


Figure 9: Recommending Function Service Composition

Figure 9 shows the GUI of the GEODISE WCE in which the sequence of yellow boxes stands for a workflow construction process and the cascading popup menus demonstrates the recommending mechanism. The recommender system runs in the background. When activated, the recommender system monitors the WCE workspace. Each time a function

from the left-hand side panel is selected and dropped into the composition area, the recommender system 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; and then it 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.

A workflow built in this way can inherit SMD from embedded component functions. More than this, mechanisms are available to attach overall SMD to generated workflows. Therefore, WCE will not only consume functions but also generate and archive workflows' SMD into instance stores for reuse.

**Function level advice:** With rich SMD embedded in function descriptions, users can obtain configuration information by following ontological links, for example, what are the types 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 a workflow construction process in a textual Domain Script Editor (DSE). The recommender system running in backend can suggest multiple choices for function configuration, and accomplish auto-completions and advise alternative values to assist textual workflow construction.

The recommender system provides an effective way to reuse services via their SMD. It can be applied to any domain as long as a semantically enriched service 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 tracing and trust.

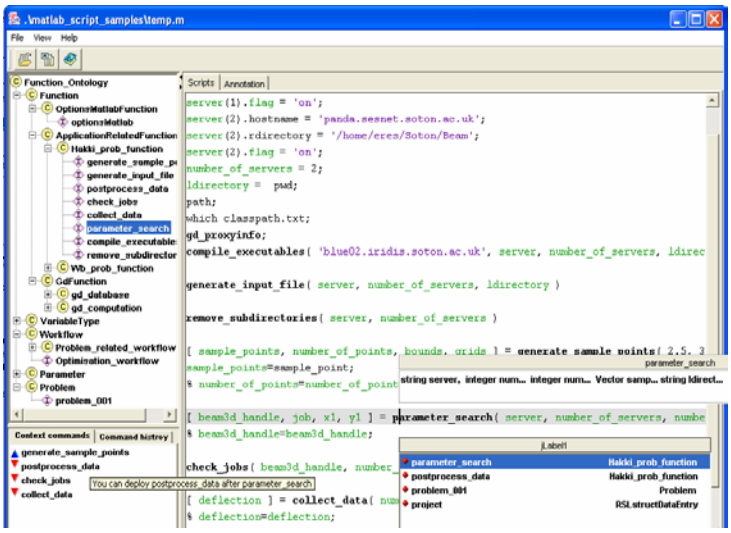


Figure 10: Recommending Function Service Configuration

# RELATED WORK

The Monitoring and Discovery Services (MDS) in the Globus toolkit and the Relational Grid Monitoring Architecture (RGMA) developed in the European DataGrid are two main information systems for Grid services. Both of them make use of metadata for publishing and discovering service status and configuration information. MDS is based on Globus' Grid Information Indexing Service (GIIS) (www.globus.org/research/papers/MDS-HPDC.pdf) and Grid Service Information Service (GRIS) (www.globus.org/mds/extending-gris.html). R-GMA is based on the GGF's Consumer-Producer model and the relational database model. MDS and RGMA mainly focus on low-level service information needed to run jobs in computers, which include available memory, free disk space, network interconnection, current workload, and operating system and CPU configuration. As the terminology and its meaning used to describe these services have been mostly agreed, both MDS and RGMA have substantial commonly agreed semantic information.

In comparison, our work is different from MDS and RGMA metadata systems in two main aspects: First we target application level services, which are usually application specific, domain-dependent and consist of heterogeneous information. Second we try to capture the formal context

of service metadata, thus making the knowledge and semantics explicit. These differences also determine that the metadata representation, storage and use between our approach and MDS and RGMA are significantly different.

In Web Service area, a service's metadata, i.e. its capabilities and interfaces, is published and stored in a UDDI (Universal Description, Discovery, and Integration) repository. The repository can be queried to discover services that match with specified capabilities and attributes. Query results are returned in the form of URIs that point to the metadata (usually in the form of a WSDL document) of each service satisfying the query. The basic problems with UDDI registry are that (1) UDDI consists of no semantic information about services, it is hard to undertake semantic-based service discovery, thus ensuring the right service being found; (2) UDDI data model is business oriented, extending the data structure to support the various domains of scientific computing may cause incompatibility problems; and (3) UDDI registry is an industrial initiative, any changes need the approval and support from main industrial partners. Therefore, although the approach adopted in UDDI is a useful one, the current implementations and availability of this technology may be limiting for supporting Grid services.

Work on generating and managing semantic metadata for Web/Grid services has been investigated in myGrid project (www.myGrid.org.uk/) and METEOR-S project (lsdis.cs.uga.edu/projects/meteor-s). Both projects adopt the approach of extending existing Web Service standards such as WSDL and UDDI to provide semantic support for service automation. The focus of METEOR-S project is to add semantics to WSDL via annotation and UDDI through a service's tModel structure. While myGrid tackles these same issues by extending a service UDDI tModel, it places special emphasis on attaching structured and unstructured task/user-specific metadata. Such metadata can be published not only by service providers but also by third-party users.

While we share the similar goals as the myGrid and METEROR-S, we have adopted different approaches. Rather than extending current Web Service standards our approach builds on the latest ontology-based service description framework DAML-S and the idea of formal metadata modeling using ontologies. DAML-S service ontology has three different kinds of classes, ServiceProfile, ServiceModel and ServiceGrounding by which DAML-S can advertise the functional, i.e. what the service does, and operational, i.e. how it works and how it can be accessed, characteristics of a Web Service. Using DAML-S together with the ontology-based formal metadata modeling, the properties and capabilities of Web Services can be described unambiguously to enable automated service discovery, composition, execution and monitoring.

Our framework is also different from myGrid and METEROR-S in semantic description representation and related query mechanisms. In METEROR-S metadata is embedded in WSDL files and UDDI data structures, and represented in XML syntax. The semantics of a service description is extracted by mapping data constructs in WSDL/UDDI to corresponding concepts in ontologies. In myGrid all the information of a service is represented in RDF and stored in a triple store. Interfaces have been provided to allow users to query the service directory using the RDQL query language. In GEODISE we use DAML_OIL ontology language to represent semantic information. As DAML extends the data constructs of RDF Schema it has more expressive capability for knowledge representation. DAML_OIL is based on description logic, thus it enables description-based reasoning and classification.

Research on semantic metadata for Grid services is also conducted in individual areas such as semantic service description (Johnston 2004), discovery (Sirin, Parsia et al. 2004) and composition (Chen, Shadbolt et al. 2003) (Medjahed, Bouguettaya et al. 2003). However, there is little effort towards a systematic and integrated approach to managing service 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 as developed in this paper.

# CONCLUSIONS

Creating and populating rich semantic metadata on the Web/Grid have been commonly accepted as the route leading to the Semantic Web/Grid vision. Semantic Web/Grid services are key research threads towards this holy-grail. This paper describes our effort towards the next generation service-oriented computing infrastructure with rich metadata and semantic support. It presents an integrated framework for SMD management for Web/Grid services, which is based on the latest ontology and the Semantic Web technologies coupled with a service-oriented computing paradigm. Issues related to the lifecycle of service SMD management such as SMD modeling, generation, storage and reuse, are analyzed and discussed, solutions are proposed and a suite of tools, APIs and mechanisms are developed to support SMD management, which form the backbone of the Semantic Wes/Grid infrastructure.

The approach has been applied to SMD management in GEODISE e-Science project. We have generated SMD for EDSO services and stored them in a repository. We have carried out SMD-based service discovery using DL-based query services. We also develop the recommender system to demonstrate the deep use of SMD, which provides advice on service composition. The successful integration and use of SMD with GEODISE application systems have made Grid-enabled EDSO easier and quicker. This further proves that the approach is promising. Our future work will focus on the use of natural language processing and information extraction techniques for automatic metadata collection and attachment. We shall also investigate the role and use of SMD to create provenance and trustworthiness in service-oriented computing.

# 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) and AKT (GR/N15764/01(P)).

## REFERENCES

Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web, Scientific American, 284(5), 34-43.

Broekstra, J., Kampman, A., van Harmelen, F. (2002). Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema, Proceedings of the International Semantic Web Conference 2002, Sardinia, Italy, June 10-12, 2002, 54-68.

Chen, L., Cox, S. J., Goble, C., Keane, A.J., Roberts, A., Shadbolt, N. R., Smart, P., Tao, F. (2002). Engineering Knowledge for Engineering Grid Applications, Proceedings of Euroweb 2002 Conference, The Web and the GRID: from e-science to e-business, Oxford, UK, December 17-18, 2002, 12-25.

Chen, L., Cox, S.J., Tao, F., Shadbolt, N.R., Goble, C., Puleston C., (2004). Empowering Resource Providers to Build the Semantic Grid. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Beijing, China, September 20-24, 2004, 271-278.

Chen, L., Shadbolt, N. R., Goble, C., Tao, F., Cox, S. J., Puleston C., Smart P. R., (2003). Towards a knowledge-based Approach to Semantic Service Composition, Lecture Notes in Computer Science, No.2870, 319-334.

De Roure, D., Jennings, N., Shadbolt, N. (2003). The Semantic Grid: A Future e-Science Infrastructure, Berman, F. et al ed. Grid Computing: Making the Global Infrastructure a Reality, 437-470.

Eres, M.H., Pound, G.E., Jiao, Z., Wason, J.L., Xu, F., Keane,A.J., Cox, S.J. (2005). Implementation and utilisation of a Grid-enabled problem solving environment in Matlab, Future Generation Computer Systems, 21(6), 920-929.

Foster, I., Kesselman, C. (2004). The Grid: Blueprint for a New Computing Infrastructure. 2nd Edition, Morgan Kaufmann, ISBN**:** 1-55860-933-4.

Foster, I., Kesselman, C., Nick, J., Tuecke, S. (2002). Grid Services for Distributed System Integration, Computer, 35(6), 37-46.

Goble, A.C., De Roure, D., Shadbolt, N.R., Fernandes, A.A.A., (2004). Enhancing Services and Applications with Knowledge and Semantics, 2nd Edition, Foster I. and Kesselman C., eds., The Grid: Blueprint for a New Computing Infrastructure, 431-458.

Haarslev, V., M¨oller, R. (2001). High performance reasoning with very large knowledge bases: A practical case study. Proceedings of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI2001), Washington, USA, August 4-10, 2001, 161-168.

Haarslev, V., Möller, R. (2003). Racer: A Core Inference Engine for the Semantic Web, Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), Florida, USA, October 20-23, 2003, 27-36.

Harris, S., Gibbins, N. (2003). 3store: Efficient Bulk RDF Storage. Proceedings of 1st International Workshop on Practical and Scalable Semantic Systems, Florida, USA, October 20-23, 2003, 1-15.

Hey, T., Trefethen, A.E. (2003). The Data Deluge: An e-Science Perspective. Berman, F. et al ed. Grid Computing: Making the Global Infrastructure a Reality, ISBN: 0470853190, 809-824.

Horrocks, I., Li, L., Turi, D., Bechhofer, S. (2004). The instance store: DL reasoning with large numbers of individuals, Proceedings of the 2004 Description Logic Workshop, BC, Canada, June 6-8, 2004, 31-40.

Horrocks, I., Sattler, U., Tobies, S. (1999). Practical reasoning for expressive description logics, Lecture Notes in Artificial Intelligence, No.1705, 161-180.

Johnston, W. (2004). Semantic Services for Grid-Based, Large-Scale Science, IEEE Intelligent Systems, l19(1), 34-39.

Medjahed, B., Bouguettaya, A., Elmagarmid, A. K. (2003). Composing Web services on the Semantic Web, Journal of VLDB, 12(4), 333-351.

Schreiber, S., Akkermans, H., Anjewierden, A., Hoog, R., Shadbolt, N. (1999). Knowledge Engineering and Management, the MIT Press, London.

Sirin, E.; Parsia, B.; Hendler, J. (2004). Filtering and selecting semantic Web Services with interactive composition techniques, IEEE Intelligent Systems, 19(4), 42-49.

Zhuge H. (2005) Semantic Grid: scientific issues, infrastructure, and methodology, ACM Communication 48(4): 117-119.

Zhuge, H. (2004). The Knowledge Grid, World Scientific Publishing Co., Singapore.

## ABOUT THE AUTHORS

**Liming Chen** is a lecturer at the School of Computing and Mathematics, University of Ulster. He received his BSc and MSc degrees in Software Engineering from Beijing Institute of Technology, PR China in 1985 and 1988, respectively. He obtained his PhD degree in Artificial Intelligence (intelligent agent) at De Montfort University, UK in 2002. He had worked on the UK E-Science project GEODISE and the EU FP6 IST project PROVENANCE in the School of Electronics and Computer Science at the University of Southampton before he joined the University of Ulster. His research interests are the Semantic Web, Semantic Grid, Semantic Web Services, information/knowledge fusion and reasoning, intelligent recommender systems, agent technologies and their applications in e-science and e-business.

**Nigel Shadbolt** is a Professor of Artificial Intelligence in the School of Electronics and Computer Science at the University of Southampton. He is a Fellow and Deputy President of the British Computer Society and chairs the Society's Knowledge Services Board. He is the Director of the EPSRC Interdisciplinary Research Collaboration in Advanced Knowledge Technologies (AKT). From 2001 to 2004 he was Editor in Chief of IEEE Intelligent Systems and in 2005 he was appointed Emeritus Editor in Chief. He also serves as a senior member of various national and international organisations. His research concentrates on two ends of the spectrum of AI – namely, Knowledge Technologies and Biorobotics.

**Carole Goble** is a Professor in the School of Computer Science in the University of Manchester. She is a co-director of the Northwest Regional e-Science Centre and the Information Management Group, and the director of UK e-Science myGrid project. She is also Editor-in-Chief and manager of the Journal of Web Semantics: Science, Services and Agents on the World Wide Web. She also serves as a senior member of various national and international organizations. Her research has been focused on the Semantic Web, e-Science, the Grid and in particular the Semantic Grid.

**Feng Tao** is a research fellow in ECS, University of Southampton. After finishing his PhD on data mining and knowledge discovery, he worked in Southampton regional E-Science centre where his contribution was mainly on application of Semantic Web and Knowledge Technologies in E-Science and the Grid, in particular the Engineering Design Search and Optimisation. His research interests include data mining, ontology, semantic web, knowledge management, agent technology in the fields of e-science and e-learning.