Submitted to IEEE/WIC International Conference on Web Intelligence (WI2004)/Intelligennt Agent Technology
20-24, September, 2004
Beijing, China

Empowering Resource Providers to Build the Semantic Grid

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

Abstract

The future success of Grid-enabled e-Science depends on the availability of semantic/knowledge-rich resources on the Grid, i.e., the so-called semantic Grid. This requires not only novel knowledge modelling and representation methods but also a shift of knowledge acquisition and population from a limited number of specialised knowledge engineers to resource providers. To this end we have developed a lightweight ontology-enabled tool, Function Annotator, to support resource providers in capturing and publishing resource semantics and knowledge. Function Annotator takes a different line to most knowledge acquisition tools in that it is designed for use by resource providers, probably in the absence of a knowledge engineer. Its aim is to facilitate large scale knowledge population on the Grid. Function Annotator is built on the state of the art of semantic web technologies, such as ontology, OWL language, instance store and DL-based reasoning, thus ensuring flexibility and scalability on the Grid. This paper describes the tool's role in a Grid-oriented resource lifecycle, its underlying technologies and implementation. It also illustrates the usage of the tool in the context of engineering design search and optimisation.

1. Introduction

One of the core challenges in knowledge management lifecycle and also in knowledge intensive system development is knowledge acquisition and modelling, i.e. the elicitation, collection, analysis, modelling and validation of domain knowledge by interaction and collaboration between knowledge engineers and domain experts. To improve the process of knowledge acquisition (KA) researchers have developed a number of principles, methodologies [1] [2], techniques and tools [3] over years. These approaches and techniques, however, are usually efficient and effective for standalone and one-off applications. For example, knowledge models such as ladders, network diagrams and tabular, and knowledge representations such as unified modelling language (UML), concept modelling language (CML) and HTML[1] are usually application specific and also closely related to the underlying reasoning mechanism. In most cases knowledge acquired for one application cannot be used for another. Furthermore current KA tools are knowledge engineer-oriented. Domain experts are not able to use these tools to capture knowledge.

There is currently vigorous research into e-Science [4] (large scale science over the Internet via the sharing and coordinated use of diverse resources in dynamic, distributed virtual organisations) and the Grid [5] (the fundamental computing infrastructure to support the vision of e-Science). Whilst a whole raft of Grid middleware [6] [7] that provides core Grid functionality and a diversity of Grid applications [8] [9] are being developed, work so far has shown that large scale semantic/knowledge support on the Grid, i.e., the realisation of the Semantic Grid [10], will be an important component in delivering the true potential of e-Science.

Scientific computing in e-Science usually involves very complicated processes. Each process is normally composed of many steps of computation. Each scientific computation is a resource that may come from different organizations and most probably is represented by

1 HTML, XML, RDF, RDFS and OWL that will be mentioned later are all W3C standards at http://www.w3.org/.

different models and terminologies. In addition, different scientific disciplines have different problems, each dependent on different aspects of domain-specific knowledge. It is usually the case that resources required for each scientific activity is domain specific and problem dependent. Obviously an effective realization of the Grid computing paradigm, namely to promote the seamless integration of highly flexible and distributed coalitions of resources, requires that resource providers should expose resources with explicitly represented knowledge in common knowledge models so that resources are used to the best effect.

An ontology has been generally regarded as a formalised representation of the knowledge in a domain taken from a particular perspective or conceptualisation. It can be used to share and communicate knowledge, both between people and between computer systems. With an ontology as a common knowledge-preserving structure we still need the means to acquire and attach knowledge to these resources before putting them on the Grid. Traditionally such tasks are accomplished by knowledge engineers and domain experts on a case-by-case basis. Given the scale and diversity of e-science and also the huge demand on knowledge support, the observations are that (1) the population of knowledge-rich resources on the Grid should become a constant activity and routine practice performed by as many resource providers as possible; (2) conventional knowledge acquisition and modelling practices cannot meet the requirements of e-Science, i.e. the dynamic, collective and just-in-time/ real-time publishing of knowledge–rich resources. Resource providers should be mobilised and given central roles to participate in the process. We suggest that KA tools oriented towards resource providers should be developed so that resource providers are better equipped to fulfil the responsibility of acquiring and publishing knowledge-rich resources.

In this paper we first give the motivation for developing resource provider-oriented KA tools based on our research experience on Geodise [11]. In section 3 we introduce a Grid-oriented resource supply and consumption lifecycle and the roles of resource providers. Section 4 details key underlying technologies for the tool. We describe Function Annotator implementation and its usage scenario in section 5. Section 6 discusses related work in related communities. We conclude the paper and discuss future work in section 7.

2. Motivations

Developing a tool for resource providers to conduct KA is inspired and driven directly by our work and experience on one of UK e-Science pilot - Grid enabled optimisation and design search in engineering (Geodise) [11]. Geodise 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 application and analysis codes, and distributed computing and data resources on the Grid. Engineering design has been practised for many years, which has accumulated huge, valuable resources including computational algorithms, design patterns and rationale, specific design prototypes, etc. Large engineering enterprises and research institutions may have data/knowledge bases of current and past products and their design information. New designs such as engines or ships are not generated from scratch but based on design data from previous practices. However, these resources are mostly so far either stored in enterprises' own archives or reside in domain experts' head. These resources are geographically located and represented in heterogeneous formats with little information on their sources and usage, which is only human-accessible and consumable (probably only the resource provider).

To make the vision of e-Science become true, two fundamental issues are: (1) how to model and represent knowledge so that it can be explicitly exposed and further shared and used on the Grid; (2) how knowledge acquisition can be done in a way that will facilitate the large-scale deployment of knowledge-rich resources on the Grid.

In Geodise we have conducted extensive knowledge acquisition. Captured knowledge is modelled in production-rules and frame-like structures and represented in various format including CML, HTML and XML. Clearly the models and representations fail to address the above issues. Therefore we have turned to Semantic Web technologies, in particular ontologies, for help. An ontology is an explicit, shared specification of the various conceptualisations in a problem domain. It contains a shared vocabulary used to describe domain concepts and the relationships among them. Ontology provides common knowledge templates/structures for a domain and ontology languages such as DAML+OIL [22] and OWL provide knowledge representation formalism.

Although we have used the latest integrated, feature-rich knowledge engineering tools such as PC-PACK [3] for knowledge acquisition, both knowledge engineers and domain experts realise that a lightweight, resource provided-oriented KA tool is necessary. The reasons are:

1). Grid computing and resource sharing will become pervasive in the near future, which requires the population of knowledge-rich resources on the Grid. It is impossible for knowledge engineers to be everywhere or to capture scientific knowledge in the right place in the right time. The right way should be for scientists to capture and model knowledge when they carry out scientific activities.

2). Scientific knowledge such as that involved in design search and optimisation is still largely the province of human domain experts expressed in the medium of natural languages. This means that formal knowledge capture processes are invaluable for editing and structuring domain knowledge. Domain experts would like to be in control, i.e. what resources are worthy publishing, and have flexibility, i.e. when to do it.

3. Resource lifecycle on the Grid

Traditionally resources are generated by resource users for their own consumption. These resources are usually either discarded or archived somewhere after use. Stored resources are in most cases only accessible and consumable by resource creators. Resource lifecycle is short and limited to the resource provider. Grid computing is all about resource reuse and sharing. This has significantly changed the scope and expectancy of a resource lifecycle. Figure 1 shows a Grid-oriented resource lifecycle that enables resource reuse and sharing through semantic metadata support. In this lifecycle a resource provide will play multiple roles, which include to:

1). Provide raw resources. Raw resources, here contained in Raw Resources component, could be previously written or newly generated algorithms and/or design solutions.

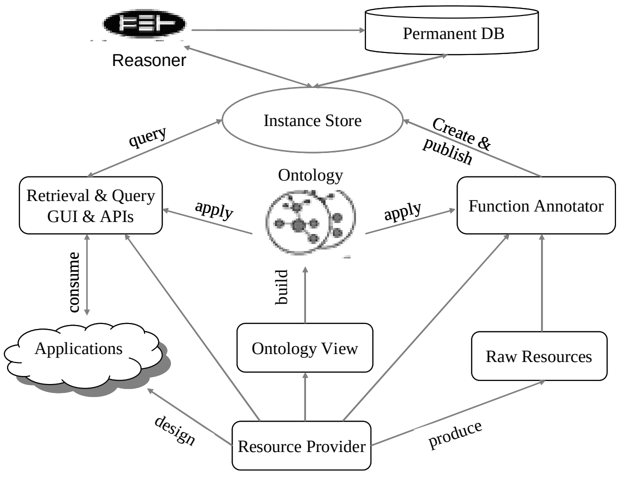


Figure 1: Grid-oriented resource lifecycle

2). Modify and edit resource knowledge models, i.e. the ontology. Resource providers are not knowledge engineers or ontologists so it is knowledge engineers' responsibility to conceptualise problem domains and develop relevant ontologies in the first stance. These ontologies will then be evaluated and verified by domain experts. During this process resource providers will obtain and be given primary knowledge about how to model their resources (concepts, properties, instances) in ontologies. An easy of use tool such as Ontology View [12] is then provided for resource providers to edit ontologies when new resources are available and worthy modelling in the ontology.

3). Elicit knowledge and put it into knowledge models, i.e. creating instances for ontological concepts. Knowledge engineers will provide resource providers a tool such as Function Annotator for them to capture and publish resource knowledge. As resource providers know exactly what knowledge is important and where it is, it is relatively easy and flexible, for example when a new resource is developed, for them to carry out the work. In such case the main role of knowledge engineer is to develop a knowledge acquisition tool for resource providers.

4). Publish the knowledge-rich descriptions of resources into central or distributed knowledge repository. Once again knowledge engineers will develop underlying infrastructure such as instance store [13] that has built-in physical storage medium and reasoning capability. Resource providers only need to use APIs or GUIs to perform archiving, retrieval and/or query.

5). Consume exposed resources in applications. With the above work done, resource providers are able to share, reuse and/or further explore available resources for problem solving.

The compelling feature of the above lifecycle is that knowledge engineers' role in knowledge acquisition and modelling shifted to the provision of underlying infrastructure. On the other hands, resource providers are given more control and flexibility in that they can conduct knowledge modelling and elicitation whenever necessary and populate knowledge repository on the Grid with knowledge-rich content in a large scale that would not be possible ever before.

4. Function annotator design

Knowledge acquisition is often viewed as a very complex task requiring dedicated knowledge engineering expertise. To empower resource providers to capture and publish resource-related knowledge, we have developed Function Annotator, a lightweight knowledge acquisition tool. Function Annotator is based on a new knowledge engineering framework, i.e. separating knowledge modelling from knowledge elicitation. The aims of such a framework are, on one hand, to provide commonly shared, domain specific but application independent knowledge models for use by a community on the Grid. On the other hand, it allows for not only knowledge engineers but ordinary resource providers to capture knowledge by simply filling in the knowledge models. Function Annotator tries to further simplify the process of knowledge acquisition by providing a number of hand-on supports that are particularly needed for resource providers. Below we describe the enabling technologies for Function Annotator.

4.1. Web-oriented knowledge modelling

To expose, share and reuse knowledge-rich resources on the Grid the fundamental question is how knowledge preserving templates/structures and the knowledge they contain are modelled in an appropriate way so that other users can recognise and extract embedded knowledge. To resolve this issue, we have adopted ontologies as knowledge models for holding knowledge for distributed resources.

Ontologies are different from other knowledge models in that they contain commonly agreed knowledge structures, i.e. domain concepts and the relations among them, and also shared terminology for describing these knowledge structures. This means that resource providers, no matter where they are, can use these same structures to preserve, publish knowledge-rich resources and equally consume resources from other experts. Ontologies provide a common medium for inter-agent information transfer, which is applicable to both humans and machines.

4.2. Resource knowledge representation

Traditional knowledge representation formalisms are not intended to deal with the sharing and interoperability of knowledge in a distributed environment such as the Grid. Therefore, it is natural and straightforward to adopt the web ontology language (OWL) as the underlying knowledge representation language.

OWL builds upon existing Web standards such as XML, RDF and RDFS. The resource description framework (RDF) is a language for representing information about resources in the Web. It is particularly intended for representing metadata about Web resources, such as the author, functionality, and modification history, copyright and licensing information about a algorithm and/or software tool. RDF Schema (RDFs) defines a vocabulary (terms) for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes. On top of RDF and RDFs OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. This empowers OWL to be able to represent knowledge about complicated resources that are usually used in scientific activities.

OWL is also different from RDF(s) in that it is underpinned by the expressive description logic. Therefore, OWL-represented knowledge can support sophisticated reasoning and knowledge extraction. Any DL-based reasoner such as [14][15] can perform various operations such as concept classification, subsumption checking, retrieving definitional information, navigating concept hierarchies, and retrieving lexical information against knowledge against OWL-based knowledge repository. Description-based reasoning capability has many applications. For example, it can be used to discover required resources via a concept match-maker or to retrieve sets of ontological concepts matching some arbitrarily defined queries through classification and subsumption reasoning.

4.3. Knowledge storage

There are three different mechanisms to store knowledge about a resource. First it can be added into the original resource with an embedded piece of descriptions. Semantic web community usually uses this approach to attach semantics to web pages. Second, knowledge can be saved in a separate file in the same location as the resource. Thirdly knowledge can be archived in a central knowledge repository separate from resources. In the context of Grid computing it is supposed that resources are owned by dynamic virtual organisations and geographically located. These resources 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 we decided to build a central knowledge repository for distributed resources, which can also serve as a registry service.

When Grid resources are modelled using ontologies and represented in OWL, knowledge objects will be generated as OWL individuals that are independent of the original resource formats and/or providers. In such cases both knowledge objects and knowledge structures will be save together in an ontology file that is actually a knowledge base. Applications can then consume knowledge by accessing the ontology file and carrying out DL reasoning over individuals. Unfortunately, existing technologies, either Racer's assertion reasoning [15] or FaCT's terminological reasoning over pseudo-concepts [14], fail to scale up to the size of over 100,000s individuals that is usually required by real scientific applications on the Grid.

We have developed the instance store technology [13] to tackle this problem. The instance store uses a relational database such as MySQL and Oracle as permanent storage media and a DL reasoner to support reasoning. This means that assertions over individuals are stored in a database, together with information inferred using a DL 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 that database and subsumption and classification requests to the reasoner.

4.4. Ontology view

One observation from Geodise is that resource providers express strong desire to have control and flexibility to edit ontologies once they are built by knowledge engineers. While there are many tools available for building ontologies such as OilEd [16] and Protégé [17], they are all intended for knowledge engineers to use. Their rich functionality and complex GUIs usually scare away resource providers from using them.

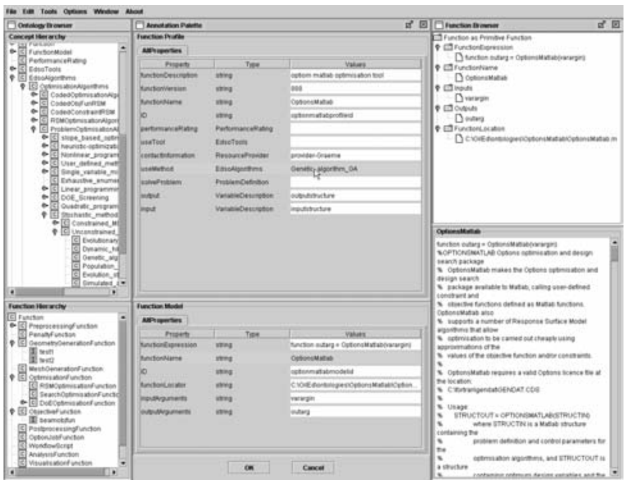
Ontology view is a lightweight ontology editor specially designed for resource providers to create knowledge models when new resources are available and modelled. Full detail about Ontology View is beyond the scope of this paper.

5. Implementation and usage

We have applied Function Annotator and its underlying framework to Geodise. As Geodise uses Matlab as its problem solving environment [18], i.e. to access and execute Grid resources in Matlab, Geodise resources, including optimisation algorithms, compute, database and application tools, at the moment are all Matlab functions rather than web/Grid services. Matlab has two types of functions, primitive functions and scripts. A primitive function has its specific interface and the function body can be treated as a blackbox. A Matlab script usually consists of a number of primitive functions. Scripts have no explicitly defined interface and cannot be treated as a blackbox. A script is actually a workflow or sub-workflow for a problem. Below we describe in details Function Annotator implementation and its usage scenario in the context of Geodise.

5.1 Implementation

We have implemented Function Annotator using OWL APIs. Figure 2 shows the GUI of the tool, which consists of an Ontology Browser, an Annotation Palette and a Function Browser. The left hand panel is the Ontology Browser that contains a concept hierarchy and a function hierarchy. The concept hierarchy presents a list of knowledge structures for a problem domain. It should be domain specific, for example the function ontology in engineering design search and optimisation (EDSO). It is used for resource providers to browse available knowledge structures so they can choose suitable ones for knowledge acquisition. If a required structure is not available, users can add the model by editing the ontology through Ontology View. The function hierarchy displays available knowledge objects under different function categories. These knowledge objects will be retrieved from backend knowledge repository on user's demand. They can be modified, edited such as cloning, and reused to generate new knowledge objects.



Property	Type	Values
functionDescription	string	options matlab optimisation tool
functionVersion	string	000
functionName	string	OptionsMatlab
ID	string	optionmatlabprofileid
performanceRating	PerformanceRating	
useTool	EdsoTools	
contactInformation	ResourceProvider	provider-Graeme
useMethod	EdsoAlgorithms	Genetic_algorithm_GA
solveProblem	ProblemDefinition	
output	VariableDescription	outputstructure
input	VariableDescription	inputstructure

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

Figure 2: Function annotator GUI

The right hand panel is the Function Browser, which is used to load Grid resources for knowledge acquisition. As the main resources in Geodise are Matlab functions, Function Annotator particularly targets dealing with them. We have provided parsing capability to facilitate automatic information extraction based on function interface. As primitive Matlab functions have conventions for interface specification, we are able to obtain important information about a resource such as input, output parameters and location details directly from original resource sources. The extracted information will be listed on the top window of the right hand panel. It is usually enough for resource providers to fill in relevant knowledge structures in the Annotation Palette. For Matlab scripts information about input, output and contained primitive functions are all embedded in the script body. Apart from automatically extracted information it is necessary to look into the content of a script to get detailed information. In such cases resource providers need to do more manual knowledge capture in order to instantiate a knowledge structure.

The Annotation Palette in the middle of the GUI is where knowledge acquisition takes place. It consists of two panels, i.e. the Function Profile at the top and the Function Model at the bottom. Function Profile contains two types of knowledge. One is metadata about the function such as authors, version, methods used, preconditions required etc. These metadata are specified using formal ontological concepts. The second type of knowledge is about the semantics of the function input/output interface. Function Model is used to hold information on how the function works and be invocated. This includes input/output arguments, location and expression. For scripts it could contain information on embedded functions as well as their sequential details.

Using Function Annotator the KA process is as follows:

- Resource providers load resources into Function Browser, for example, Geodise functions.
- Resource providers select appropriate knowledge structures from the ontology for the function. This can be done by navigating the concept tree of the ontology in the left hand panel. Alternatively users can follow the popup selection choices powered by the ontology.
- Once knowledge structures are selected, ontology-driven forms will be automatically generated in the Annotation Palette for resource providers to fill in relevant information.
- Resource providers can fill in forms in different ways in terms of the knowledge sources. In addition to direct input, resource providers will often drag and drop the extracted interface information into forms in Annotation Palette from the Function Browser. They can also drag and drop knowledge structure and/or knowledge objects from the Ontology Browser. For Matlab scripts, resource providers may need to markup texts representing information embedded inside resource sources in Function Browser, then copy and paste them into forms in Annotation Palette.
- Knowledge objects will be first represented as OWL individuals in memory in OWL language, and when archiving, translated into instances in instance store in DIG format.

Using Function Annotator resource knowledge can be captured at multi-level of abstraction. For example it could be a high-level concept/type, an instantiated knowledge objects or concrete values. Resource providers can add information into knowledge structures as detailed as they know or just give simple or general information on some aspects of a resource if they think it is not important. In summary: with Function Annotator, knowledge acquisition can be done to the exact degree necessary.

5.2 Usage scenario in Geodise

Figure 3 shows Function Annotator usage scenario by taking a broader view of its application in Geodise. In this scenario the first step is function characterisation. This includes function classification and categorisation, function interface analysis (input/output modelling) and terminology extraction. A domain ontology is then built based on function characterisation. The domain ontology will serve as a set of explicit knowledge structures for preserving knowledge. In addition to the domain ontology, a upper level function ontology will also be provided to enable the addition of metadata to resources. These metadata could contain any arbitrary information that can help use the resources, mostly those such as authors, copyright, version, licences and generic function information. Metadata is extremely important for resource discovery, sharing and use regarding provenance and trust.

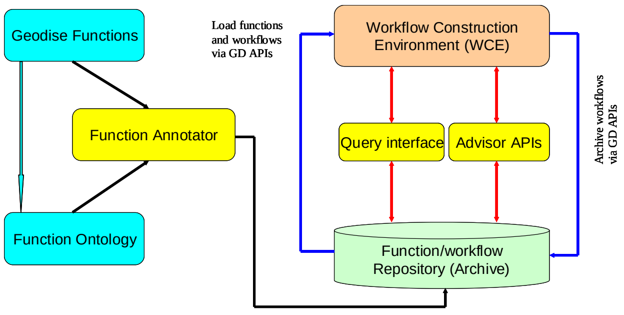

Geodise Functions
Load functions and workflows via GD APIs
Workflow Construction Environment (WCE)
Function Annotator
Query interface
Advisor APIs
Archive workflows via GD APIs
Function Ontology
Function/workflow Repository (Archive)

Figure 3: Usage scenario in Geodise

Following the above steps users (mainly resource providers) can use Function Annotator to capture embedded knowledge within functions as described above. The acquired knowledge independent of the original functions will be archived in the knowledge repository – the instance store. Once instance store is populated with large amount of knowledge objects, it is ready for use for knowledge intensive applications.

In Geodise a workflow construction environment (WCE) has been developed to help engineers construct a workflow. A workflow in engineering design search and optimisation is actually a solution to a specific EDSO problem. The problem solving process amounts to the process of constructing and executing a workflow. However, building a workflow for a problem requires a lot of domain knowledge, for example, what algorithm is suitable, who has this codes, how to specify the control parameters, etc. Therefore, WCE has been provided knowledge support. As can be seen in Figure 3, WCE uses high level GD APIs that build on top of OWL APIs to load functions and workflows from instance store. As knowledge regarding these functions has been explicitly modelled and represented, engineers are able to get hold of the right functions for a specific problem. To assist workflow construction a DL-based query GUI and a semantics-based workflow construction advisor are also provided to make maximum use of the knowledge-rich functions in instance store. Both tools exploit OWL reasoning capability to find functions with required characteristics.

A workflow built in this way can inherit knowledge from embedded component functions. More than this GD APIs also provides mechanisms to attach semantic metadata to a workflow. Therefore, WCE will not only consume functions but also generate and archive knowledge-rich workflows into instance store for reuse.

6. Related work

Function Annotator is a domain specific lightweight knowledge acquisition tool. Comparing with conventional KA tools, it explicitly separates knowledge elicitation from knowledge modelling. The motivation behind this is that for Grid-enabled e-Science knowledge elicitation and publishing should take place in a very large scale by as many people as possible and knowledge models need to be commonly accessible, acceptable and recognisable, thus ensuring flexibility and scalability. Traditional KA tools are effective for standalone one-off knowledge system development but fail to scale up to Grid environment by using latest web technologies such as ontology, RDF and OWL.

Function Annotator has much in common with Protégé [17] as both of them allow for creating class and property instances to describe resources, i.e. HTML pages or Matlab functions. While Protégé has strong ontology editing capabilities Function Annotator has a target roughly similar to the instance acquisition phase in the Protégé framework. The obvious difference between Function Annotator and Protégé is that the latter does not (and has not intended to) support the particular resource settings, i.e. managing and displaying resources.

Function Annotator, to some extent, can be considered as an annotation framework, though with a different focus from those like Annotea [19] and Antomat-Annotizer [20]. Early annotation tools such as Annotea all share the idea of creating a kind of user comment about web pages. The term “annotation” here is best understood as a remark to an existing document. For example, a user of these tools might attach a note like "A stochastic optimisation method" to the name “genetic algorithm” on a web page. Like S-CREAM [21] Function Annotator uses an ontology to cast strong types to comment or descriptions regarding entity’s attributes and/or properties. While S-CREAM deals with web pages Function Annotator targets on software resources such as algorithms. From this perspective Function Annotator is much like a tool for semantic web service descriptions [23]. However, Function Annotator concentrates on one particular type of resource in engineering design search and optimisation, i.e. Matlab functions rather than well-structured web services.

Function Annotator is a resource providers-oriented KA tool that distinguishes it from most others from related communities. Function Annotator may be the first of such tools to harvest the latest semantic web technologies such as the OWL interface, instance store and DL-based reasoning to enable knowledge acquisition, publishing, storage and reasoning.

7. Conclusions

In this paper we have described Function Annotator, a specialised, ontology-enabled tool for engineers to carry out knowledge elicitation and semantic enrichment. Function Annotator is underpinned by a new knowledge acquisition framework, i.e., separating knowledge modelling from knowledge elicitation, thus enabling large scale knowledge population efficiently. We have introduced the Grid-oriented resource lifecycle in which resource providers play central role in the loop of resource provision, knowledge elicitation, publishing and reuse. We have discussed the exploitation of the latest semantic web technologies to make the tool functionally unique. We have implemented the tool and demonstrated its usage in Geodise. The tool is tuned for use in design optimisation and Matlab but it could be applied to many other types of application or problem solving environment.

Initial feedback from resource providers is encouraging, indicating a strong interest and desire to have such a tool for use. However, it also shows big differences in the requirements for functionality, usability and GUI support between knowledge engineers and resource providers. Future work on Function Annotator will include the simplification of GUI, hiding as much unnecessary ‘knowledge jargon’ as possible, the provision of hand-on assistance and the enhancement of automatic information extraction and knowledge generation.

Acknowledgement

This work is supported by the UK EPSRC Geodise e-Science pilot (GR/R67705/01). The authors gratefully acknowledge the contributions from and discussions with EPSRC projects MyGrid [24] (GR/R67743/01) and AKT [25] (GR/N15764/01(P)).

References

[1] Schreiber S., Akkermans H., Anjewierden A., Hoog R., and Shadbolt N., (1999), Knowledge Engineering and Management. The MIT Press, London.
[2] Brimble, R., Oldham, K., Callot, M. and Murton, A., (1999) MOKA: A Methodology for Developing KBE Applications, Proceedings of the 8th European Conference on Product Data Technology, pp.361-366.
[3] PC PACK Knowledge tool http://www.epistemics.co.uk/products/pcpack/

[4] Hey, T. and Trefethen, A.E. (2003). The Data Deluge: An e-Science Perspective. In Journal of Grid Computing Making the Global Infrastructure a Reality, Wiley, January 2003.
[5] Foster, I. and Kesselman, C. (1999). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann.
[6] The Globus Project. http://www.globus.org/
[7] The UNICORE Project. http://www.unicore.org/unicore.htm
[8] The EUROGRID Project, http://www.eurogrid.org/
[9] The UK e-Science Initiative, http://www.rcuk.ac.uk/escience/rclinks.shtml

[10] Roure, D., Jennings, N. and Shadbolt, N. 2001. Research Agenda for the Future Semantic Grid: A Future e-Science Infrastructure http://www.semanticgrid.org/v1.9/semgrid.pdf
[11] The Geodise Project http://www.geodise.org/
[12] Ontology view ttp://www.cs.man.ac.uk/~puleston/ontoview/OntologyViews.pdf
[13] The Instance Store, http://instancestore.man.ac.uk/
[14] Horrocks, I., Sattler, U. and Tobies, S. 1999. Practical reasoning for expressive description logics. In Lecture Notes in Artificial Intelligence, No.1705 pp.161-180. Springer-Verlag.
[15] Haarslev, V. and Möller, R. (2001), High Performance Reasoning with Very Large Knowledge Bases: A Practical Case Study, Seventeenth International Joint Conference on Artificial Intelligence, IJCAI-01, http://www.cs.concordia.ca/~haarslev/ publications/ijcai-2001.pdf
[16] Bechhofer, S., Horrocks, I., Goble, C. and Stevens, R. 2001. OilEd: a Reason-able Ontology Editor for the Semantic Web DL2001, 14th International Workshop on Description Logics, Stanford, USA. http://oiled.man.ac.uk/.
[17] Protégé ontology and knowledge editor http://protege.stanford.edu/index.html
[18] Pound, G. E, Eres, M. H, Wason, J. L, Jiao, Z, Keane, A. J.and Cox, S. J. (2003) A Grid-Enabled Problem Solving Environment (PSE) for Design Optimisation within Matlab. 17th International Parallel & Distributed Processing Symposium
[19] The Annotea Project http://www.w3.org/2001/Annotea/
[20] OntMat-Annotizer Annotation Tool http://km.aifb.uni-karlsruhe.de/annotation/ontomat.html
[21] Handschuh, S, Staab, S, and Ciravegna, F (2002), S-CREAM-Semi-automatic CREAtion of Metadata, In: Proc. of the European Conference on Knowledge Acquisition and Management - EKAW-2002.
[22] DAML+OIL http://www.daml.org.
[23] DAML Services Ontology. (2002). DAML-S: Web Service Description for the Semantic Web. In The First International Semantic Web Conference (ISWC). http://www.daml.org/services/
[24] The MyGrid Project http://mygrid.man.ac.uk/index.shtml
[25] The Advanced Knowledge Technologies (AKT) Project. http://www.aktors.org/