Towards a Knowledge-Based Approach to Semantic Service Composition

Liming Chen[1], Nigel R. Shadbolt[1], Carole Goble[2], Feng Tao[1], Simon J. Cox[3], Colin Puleston[2], and P.R. Smart[4]

[1] Department of Electronics and Computer Science, University of Southampton
Highfield, Southampton, SO17 1BJ, U.K.
{lc,nrs,ft}@ecs.soton.ac.uk
[2] Department of Computer Science, University of Manchester
Oxford Road, Manchester, M13 9PL, U.K.
{carole,colin.puleston}@cs.man.ac.uk
[3] School of Engineering Sciences, University of Southampton
Highfield, Southampton, SO17 1BJ, U.K.
sc@ecs.soton.ac.uk
[4] Epistemics Ltd., Strelley Hall, Nottingham NG8 6PE, U.K.
paul.smart@epistemics.co.uk

Abstract. The successful application of Grid and Web Service technologies to real-world problems, such as e-Science [1], requires not only the development of a common vocabulary and meta-data framework as the basis for inter-agent communication and service integration but also the access and use of a rich repository of domain-specific knowledge for problem solving. Both requirements are met by the respective outcomes of ontological and knowledge engineering initiatives. In this paper we discuss a novel, knowledge-based approach to resource synthesis (service composition), which draws on the functionality of semantic web services to represent and expose available resources. The approach we use exploits domain knowledge to guide the service composition process and provide advice on service selection and instantiation. The approach has been implemented in a prototype workflow construction environment that supports the runtime recommendation of a service solution, service discovery via semantic service descriptions, and knowledge-based configuration of selected services. The use of knowledge provides a basis for full automation of service composition via conventional planning algorithms. Workflows produced by this system can be executed through a domain-specific direct mapping mechanism or via a more fluid approach such as WSDL-based service grounding. The approach and prototype have been used to demonstrate practical benefits in the context of the Geodise initiative [2].

1 Introduction

The Grid [3] has been developed to support a vision of e-Science [1] where the sharing and coordinated use of diverse resources in dynamic, distributed virtual organizations is commonplace. Web services [4] have been designed to wrap and expose resources and provide interoperability between diverse applications. The combination of these technologies has seen Grid technologies evolving towards an

D. Fensel et al. (Eds.): ISWC 2003, LNCS 2870, pp. 319–334, 2003.

Open Grid Services Architecture (OGSA) [5] (a service-oriented distributed computing paradigm), which sees the Grid as providing an extensible set of services that virtual organizations can aggregate in various ways to solve domain specific problems.

Scientific computing in e-Science usually involves very complicated processes. A 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 in different models and terminologies. Apart from this, other features of scientific computing include: (1) different disciplines have different problems, each dependent on different aspects of domain-specific knowledge; (2) the nature of the problem being dealt with often changes as the computation process proceeds, a fixed workflow is therefore nearly impossible for all but the most simple cases; and (3) both the underlying computing resources and the information input for the process are dynamic. Obviously an effective realization of the Grid computing paradigm, namely to promote the seamless integration of highly flexible and distributed coalitions of service-oriented components, requires not only an explicit description of resources so that they can be discovered, shared, understood and processed by both humans and machines, but also domain specific knowledge about how these resources should be composed, configured and executed to achieve some problem-solving goals.

Service discovery, description and messaging have been tackled in the web services community with introduction of a set of 'industry-standard' protocols (e.g. UDDI [6], WSDL [4] and SOAP [7]). However, none of these standards defines the meaning of services and their parameters in a way that transcends the tendency of agents to use their own terms and frame of reference. Furthermore, these protocols do not address the need to coordinate the sequencing and execution of services as part of some larger information processing tasks. Some industry initiatives have been developed to address this issue, such as WSFL [8], XLANG [9] and BPEL4WS [10]; however, such initiatives generally focus on representing service compositions where the flow of the process and the bindings between the services are known *a priori*. For scientific computations the knowledge required to select and coordinate the activity of available services is usually specific to the application domain. It is often the case that resource selection cannot be specified in advance of the execution of individual components of the more global workflow specification. As such it is apparent that pre-defined service sequencing and binding is not sufficient in scientific computing contexts.

Research on resource reuse on the Web has also been done in the knowledge engineering community in a different guise such as the IBROW project [11]. In IBROW resources are organized as component libraries and their competences are described using the language UPML [12]. This language uses logical formalisms and ontologies to describe the problem solving capabilities of the components. The central idea of the IBROW project is that of brokering [13] between libraries of software components and a user. Whilst IBROW has come up with a number of approaches to library organization, component specification, broker architectures and brokering mechanisms, it is not clear where domain knowledge can fit in the generic brokering service paradigm and how that knowledge is exploited in component selection and configuration. At this point IBROW concentrates on component discovery and configuration rather than component composition for problem solving.

Recently the Semantic Web technologies [14] have been used to provide more explicit and expressive descriptions for web services [18]. The purpose of semantic service descriptions is to facilitate service discovery based on underlying semantics that is enriched by means of ontologies using ontology description languages such as DAML+OIL [15], DAML-S [16] and OWL [17]. Obviously semantic service descriptions are not intended to provide the knowledge about when and how such services should be used to solve a problem, for example in the case of service composition. Some research [19] has exploited semantic matching via ontology-driven reasoning to conduct service composition. However, for some domains there is not always a well-structured high-level ontology that can be used to characterize the domain activities. Furthermore, for complex computations there may be multiple choices for the next step of a workflow sequence. The selection and configuration of a service for a specific problem are usually dependent on rich nexuses of domain knowledge.

We argue that both semantic service descriptions and domain-specific knowledge-based decision support services are essential ingredients for resource synthesis in e-Science. Semantic service descriptions support effective service discovery, seamless resource integration and reuse on the Grid. Knowledge-based decision-making support systems can suggest what should be done next during a service composition process and which service should be chosen once a number of services are discovered. All decisions can be made dynamically by taking into consideration the problem characteristics, service performances and previous computation results. Furthermore, once a service is selected, knowledge support can be further provided for the configuration of that service. As such we contend that web-based service–oriented applications, both e-Science and e-Business, ought to exploit semantic service descriptions and domain knowledge in order to solve complex problems through automatic, seamless resource synthesis on the Web/Grid.

This paper introduces a knowledge-based approach and framework for semantic service composition. In section 2 we briefly discuss semantic resource description using the DAML-S service ontology. Section 3 describes the knowledge-based advice system, in particular, its recommendation strategy and implementation architecture. Section 4 presents the semantic service composition framework for the knowledge-based approach. An implementation prototype is given in section 5 to demonstrate the approach with respect to an example problem taken from the GEODISE initiative. We conclude in section 6 with some initial findings and possible future work.

2 Modeling Resources with Semantics

Web/Grid services are used to model resources for scientific activities, which include not only information but also assets (data storage and specialized experimental facilities), capabilities (computational systems) and knowledge (recommendation and advice). They are represented and described using the WSDL, which uses XML [20] to describe services as a set of endpoints operating on messages. The implementation of WSDL during service design is usually more concerned with the signature of a service, i.e. the identifiers of the service and its parameters. Based on this description, it is usually impossible for software agents to figure out the precise meaning of the service identifiers and functionality provided by the service. The lack of semantics in

the abstract functionality description of the service, i.e. the capabilities of the service, makes it difficult for machines to discover and use the service at the right time.

Ontological engineering plays a central role in incorporating semantics into service descriptions. An ontology is an explicit, shared specification of the various conceptualization in a problem domain. It not only provides a common language for interoperability but also adds meaning and relations to service descriptions. Ontology representation languages, such as RDF Schema [21], DAML+OIL or the ontology web language OWL, can be used to characterise the service-portfolio offered by a web service in a more expressive manner than the existing WSDL, thereby opening up the possibilty of automatic service discovery and use.

Semantically-enriched service descriptions can also be provided by the DAML Service ontology language DAML-S – a service description language that is itself written in RDF. DAML-S partitions a semantic description of a web service into three components: the service profile, process model and grounding. The Service Profile describes what a service does by specifying its inputs, outputs, preconditions, effects and other properties. The Process Model describes how a service works; each service is either an Atomic Process that is executed directly or a Composite Process that is a combination of other sub-processes. The Grounding contains the details of how an agent can access a service by specifying the details of the communication protocol, i.e. the parameters to be used in the protocol and the serialization techniques to be employed for the communication.

DAML-S allows the definition of classes of related services and can establish links to other concepts that describe specific service types and their properties. This makes service discovery much easier in terms of the built-in links, thus facilitating resource reuse. For example, in the engineering design domain a mesh generation service has a geometry file as its input, which is linked to a geometry generation service, and a mesh file as its output, which leads to the code analysis service. The mesh generation service itself uses the Gambit meshing tool as its process model.

3 A Knowledge-Based Advice System for Service Composition

Scientific activities often involve constructing a workflow either manually or automatically to realize a particular experiment or series of computations. In the service-oriented Grid computing paradigm this process amounts to discovering services on the Grid and composing those services into a workflow. Some domains such as a supermarket demand-supply chain have a fixed flow of process and stationery bindings between services. However, for most scientific disciplines a workflow is both domain-specific and problem-dependent. The appropriate selection of services at each point in the workflow often depends on the results of executing the preceding step. Moreover, the selection of a service from a set of competing services with similar capabilities is usually determined by the exact nature of the problem as well as the performances of the services available. As a result, it is not practical to specify, a priori, the precise sequence of steps for a problem goal. The successful orchestration of component services into a valid workflow specification is heavily dependent on bodies of domain knowledge as well as semantically enriched service descriptions.

A knowledge-based advice system aims to support automatic or semi-automatic service composition by providing advice constrained by bodies of domain-specific knowledge. It is described in detail below.

3.1 Strategies for Knowledge-Based Advice

There are two approaches to providing service composition advice. One is based on the semantic service descriptions, i.e. the conceptual links between services and their properties [18] [22]. It makes use of available information about service profiles such as the preconditions, constraints and outputs of the service in order to assess the potential fit of each service to a particular role in the workflow specification. The expressive description logic of DAML+OIL enables a suitable reasoning engine, such as FaCT [29], to automatically retrieve a service that matches the semantic description. External agents can use the outcome of such reasoning engines to select a service commensurate with their information processing goals. Often, however, such systems are limited with respect to the appropriate selection of services suited for a specific task or with the appropriate configuration of service parameters.

The knowledge-based approach to the provision of service composition advice can often succeed in situations where ontology-driven reasoning proves inadequate. For example, in the domain of engineering design search and optimization there are over a hundred 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. Knowledge about the correct method to choose in a particular situation as well as the appropriate configuration of method parameters is an important feature of expert-level performance and a vital ingredient of problem-solving success. Any system concerned with the appropriate selection of optimization methods, therefore requires access to an exquisitely detailed representation of the knowledge contingencies relating problem characteristics and design goals with the appropriate selection and configuration of available methods.

The knowledge-based approach builds on the classical model of knowledge-based decision support systems that make extensive use of domain knowledge. Therefore, it relies heavily on the techniques of knowledge engineering [23]. The development of knowledge-based systems usually involves (1) the identification of knowledge-intensive task areas, and the gaining of a detailed insight into the ways in which knowledge is used to yield favorable decision outcomes, (2) the elicitation of, or indirect acquisition of, domain knowledge using knowledge acquisition (KA) techniques, (3) The modeling of human-level knowledge in formal, symbolic structures and the representation of that knowledge using a range of representational formalisms, (4) The use and reuse of knowledge in the knowledge-based system to meet the user requirements, and finally (5) The update and maintenance of both the formalized knowledge and knowledge-based systems. One feature of the latest knowledge engineering methodologies, such as CommonKADS [23], and knowledge engineering tools, is that they place special emphasis on the way in which knowledge is modeled so as to promote knowledge re-use across diverse problem-solving contexts.

Knowledge-based advice systems for service composition have the advantage of providing specific advice at multiple levels of granularity during the service composition process. At the highest level, the system can help determine what kind of

service is required against a contextual backdrop that includes problem-solving goals and procedural knowledge. Once all the services that can fulfill the required function are discovered, the advice system can recommend an appropriate service, taking into account both problem characteristics and performance considerations. More specialized, in-depth advice can also be given, for example, how to initialize and configure the control parameters of a service. Such knowledge is usually only available from experienced users or domain experts.

In order to deploy, share and re-use knowledge-based advice systems in the Grid computing paradigm, the system has been developed with three important innovations. Firstly, ontologies are used as knowledge models for representing knowledge. Second, ontologies are exploited to conceptualize knowledge systems with commonly accepted vocabulary, thus facilitating knowledge sharing and re-use. Third, knowledge systems themselves are exposed as services within the service-oriented framework of the Grid.

3.2 A Service-Oriented Architecture for Knowledge-Based Advice Systems

Traditionally, knowledge intensive systems are constructed anew for each knowledge project. There is often little reuse of existing knowledge structures and problem-solving elements. The reasons for this are legion, including the diversity of domain knowledge, the close coupling of domain knowledge with reasoning processes and the different terminologies and modeling views adopted by different users for a single domain. It is obvious that the exploitation of knowledge technologies on the Web/Grid requires that these obstacles be successfully surmounted, an insight that has led to a variety of new tools, techniques and research agendas [14] [24] [25] [26].

Based on the above consideration we have developed a generic architecture for knowledge-based advice systems that is intended to operate on the Grid (see Figure 1) [27]. The architecture has three distinguishing features. The first is that it separates domain knowledge and reasoning functions into the Application Side and Knowledge Service Side respectively. The Application Side cares about the acquisition, modeling (knowledge engineer's work) and usage (end users' requirements) of domain knowledge. Knowledge services on the Service Side provide reasoning mechanisms, advice representation and communication. This feature enables the effective re-use of domain-specific knowledge across different problem-solving contexts and the application of common reasoning processes to diverse domain-specific problems. Such an approach has many advantages in terms of ease of maintenance and re-use of knowledge components.

A second feature of the architecture is its use of multiple layers. These layers enable the effective separation of reasoning, communication and representation components into the Inference, Communication and the Application Layers. The Application Layer uses domain ontologies from the Application Side to define an application-dependent state model that is then converted to a frame-like XML schema used as a placeholder for state variables. A state model contains the description of all possible factors that can potentially affect the advice delivered by the knowledge service. It holds the state space of an application on the Application Side and uses the state information as the input to the reasoning engine in the Inference Layer. The Communication Layer deals with the transmission protocols and serialization of messages between the Application Side and the Knowledge Service Side, i.e.

transmission of the XML schema of the state model and the state information requests. The Inference Layer provides a domain-independent inference capability via a reasoning engine. The availability of a domain-specific knowledge base enables the reasoning engine to drive inferential processes that operate on the state information.

A third feature of the architecture regards its use of ontologies – the web-oriented knowledge models. Not only are the state variables of an application denoted using ontology vocabularies, as discussed above, but also the axioms, facts and rules of the knowledge base are formalized with respect to the shared repository of common terms. The use of ontology enables different users and machines to share and reuse domain-specific knowledge. These features make the proposed advice system different from traditional standalone knowledge-based systems, and contribute to its acceptability in a Grid computing environment.

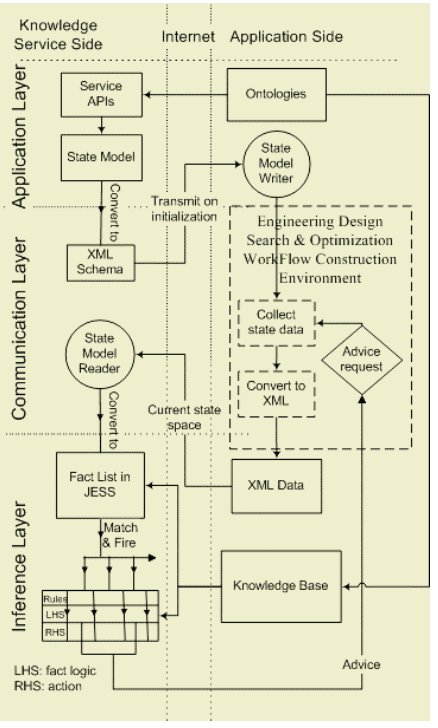

Fig. 1. The Architecture of Knowledge-based Advice System

The generic knowledge-based advice system is actually a web service, which operates as follows. The service user in the Application Side supplies domain knowledge, i.e. ontologies and knowledge bases. The knowledge service in the Knowledge Service Side creates the state model and corresponding XML schema. The state XML schema is passed onto the Application Side during knowledge service initialization. The State Model Writer in the Application Side monitors the progress of the application and collects relevant states to fill in the state XML schema. Whenever the application requests advice, the state information in the state model, i.e. an instantiated XML schema, will be sent to the knowledge service. Once the state information of the application reaches the Knowledge Service Side, it will be parsed and converted to facts. The reasoning engine in the Inference Layer will reason against these facts to provide domain-specific, context-sensitive decision support.

Figure 1 illustrates the implementation of the proposed architecture in the context of engineering design search and optimization. In this implementation, the Application Side (the user) is concerned with advice on EDSO workflow construction. Domain knowledge in this example application assumes the form of EDSO ontologies and knowledge-rich contingencies represented in a production rule-like format. The reasoning of the Inference Layer is based on JESS [28], a Java-based implementation of the CLIPS expert system shell. Outside of this domain, the aforementioned system architecture is applicable to any area of domain expertise,

providing that a suitable characterization of the domain-specific knowledge is available.

4 Knowledge-Based Service Composition Framework

We have developed and partially implemented a knowledge-based service composition framework (see Figure 2) to provide a practical demonstration of our approach. This framework uses domain knowledge and advice services to provide advice and guidance with respect to the selection, sequencing and correct configuration of services as part of constructing a workflow specification. It additionally uses semantically-enriched service descriptions to assist in the process of discovering available services for workflow specification. This ability to exploit service descriptions facilitates the workflow specification process with respect to existing descriptions of Web/Grid resources.

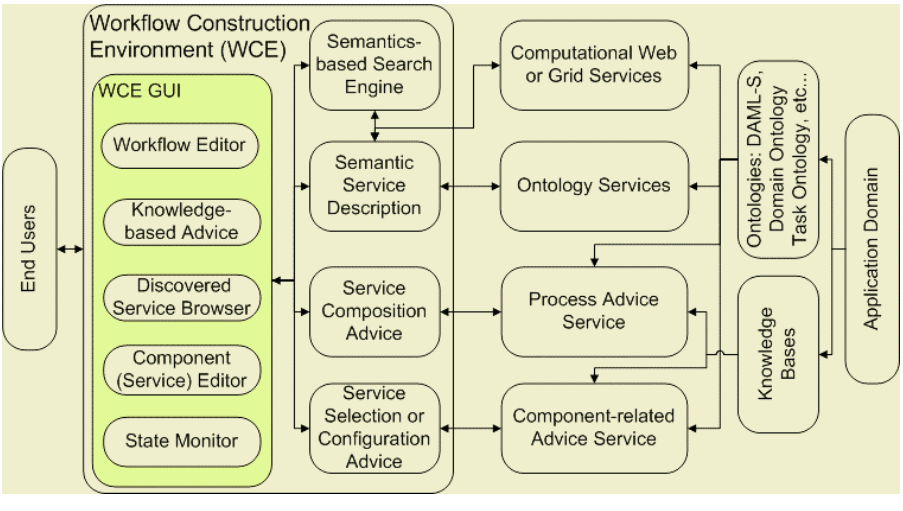

Fig. 2. The Knowledge-based Semantic Service Composition Framework

The framework consists of a set of components, mainly a Workflow Construction Environment, a set of diverse web services, Knowledge Bases and Ontologies. The Ontology component contains domain-related ontologies that provide an explicit shared conceptualization of the target domain, i.e. concepts, terms and relations. They serve as a conceptual backbone to underpin the service composition framework by supplying a common vocabulary and establishing semantic links among service properties. The ontologies were developed through ontological engineering and are exposed through the Ontology Services component. The Ontology Services provide complete access to any DAML+OIL ontology available over the Internet. Users can perform common ontological operations, such as subsumption checking, class and/or property retrieval and navigation of concept hierarchies through a set of ontology service APIs in conjunction with the FaCT reasoner [29]. Ontology Services are

implemented as standard SOAP-based web services and can be accessed, shared and re-used through the WSDL standard.

The cornerstone of the service composition framework lies in the exploitation of domain-specific knowledge contained in the knowledge bases. These knowledge bases consist of concepts, axioms and rules acquired through KAs, which conceptualize the target domain. The availability of knowledge bases is a prerequisite for advice provision during the workflow construction process.

Advice services are actually knowledge-based systems that are implemented as web services [27] such as the Process and Component-related Advice Services. They provide advice based on service requests. Users can obtain advice in two ways. First, a user may request advice according to his/her epistemic needs and requirements during the workflow construction process. Secondly, a software agent can be used to monitor the service composition process as it unfolds, and provide advice and/or recommendations along the way. Both approaches need to monitor the progress of the composition process and collect snapshots of states when advice is requested. These states are then fed into the reasoning engine to retrieve context-sensitive advice as with traditional knowledge-based systems. Advice can be provided at multiple levels of granularity, for example the process level and/or the component level, dependent on the availability of knowledge in the underlying knowledge bases.

At the core of the framework is Workflow Construction Environment (WCE) It consists of a set of WCE GUIs and tools to facilitate workflow construction. Semantic service description is undertaken using ontologies accessed via the ontology services. As the DAML-S service ontology only provides the basic schema for describing a web service, it does not provide the vocabulary with which to describe specific services in different scientific domains. Therefore, domain specific ontologies are used to incorporate domain specific functions and terminology in creating semantic service descriptions. The process of specifying semantic service descriptions is carried out in two steps. Firstly, domain ontologies, such as the task ontology and the function ontology, are created. Then, the domain specific service ontology is built using concepts from the domain ontologies. The semantic descriptions of domain-specific services are actually instances of concepts from the service ontology. Semantic service descriptions are stored in the Semantic Service Description component.

Service discovery is accomplished by the use of a Semantic-based Search Engine. It is realized through reasoners such as FaCT or MatchMaker [19] acting on the semantic descriptions of services. The services that fulfill users' requirements will be returned to users as the basis for selection in the context of workflow specification.

The WCE GUI consists of five graphical tools to assist workflow construction. Each of them presents relevant structures and information via a control panel. The Component (Service) Editor is a frame-like data-storage structure. It can be used to specify a service description for service discovery or to define a service directly by filling in the required data fields. The structure of the Component Editor is dynamically generated in accordance with the service ontology. The Discovered Service Browser displays services returned by the search engine. Users can choose a service from the panel based on the advice given for a particular workflow composition. The State Monitor monitors the workflow construction process, and collects and maintains a workflow state space. During the workflow construction process, whenever a request for advice is made, the state space can be fed into the underlying knowledge-based advice services. Advice as to what to do next and which

service should be used is then provided as output by these services. The results are shown in the Knowledge-based Advice panel. Workflows are built in the Workflow Editor where different services are connected together. Details of the workflow construction process will be described in the next section in the context of a real application.

5 Example Application: Workflow Construction in Geodise

Engineering design search and optimization (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.

Grid enabled engineering design search and optimization (Geodise) aims to aid engineers in the EDSO process by providing a range of internet-accessible web services comprising a suite of design optimization and search tools, computation packages, data, analysis and knowledge resources. A desirable feature of Geodise is that it should allow for users to compose a suite of EDSO algorithms (web services) into a workflow, i.e. to create a design solution to a specific EDSO problem. To provide such a capability we have applied our approach and the corresponding framework in Geodise. The detailed work is described below.

We have undertaken extensive knowledge and ontological engineering using CommonKADS methodology in the domain of EDSO. A substantial amount of domain knowledge has been acquired and modelled [30], for example the EDSO process knowledge in Figure 3. A number of ontologies have also been developed using OilEd [31]

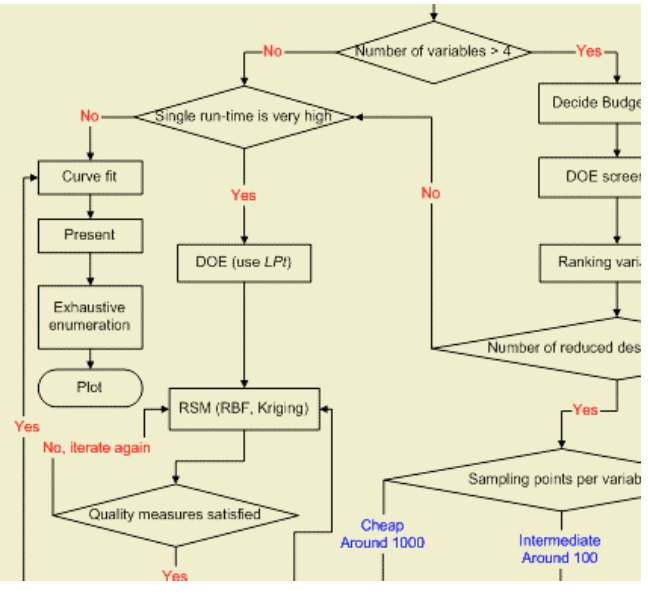

Fig. 3. Fragment of EDSO Design Process Flowchart

including the EDSO basic domain ontology, the optimization function ontology and the DAML-S-based task service ontology.

We have developed ontology services that use the emerging web ontology standard, DAML+OIL, as the underlying representation language. The services provide a set of Java APIs for common ontological operations. Ontology services are realized as a standard SOAP-based web service in Java and deployed using Apache Tomcat & Axis technologies.

We have developed a process knowledge base for EDSO design processes based on the knowledge model in Figure 3. Table 1 and Table 2 show a small subset of facts and rules in the knowledge base. We have implemented a knowledge-based advice system (discussed in 3.2) which is driven by the EDSO process knowledge [27]. The advice system monitors the design process and provides context-sensitive help at each stage of the process. This process is described in more detail below.

Table 1. Fragment of the facts in the process knowledge base

```
… …
f-6  (MAIN::resource (name "step_file") (location "d:/geodise/res/airFoilStepFile"))
f-7  (MAIN::resource (name "gambit_jou_file") (location "d:/geodise/res/gambit.jou"))
f-8  (MAIN::workflow_task (name "geometry") (input nil) (output "step_file") (relevant_commands nil) (finished? nil) (constrains nil) (dependance ))
f-9  (MAIN::workflow_task (name "mesh") (input "step_file" "gambit_jou_file") (output "mesh_file") (relevant_commands nil) (finished? nil)
(constrains nil) (dependance ))
… …
```

Table 2. Fragment of the rules in the process knowledge base

```
… …
(defrule rule1
        (not (state_panel (available_resources $?x "step_file" $?y)))        ?taskID<-(workflow_task(input $?a "step_file" $?b))
=>      (retract ?taskID)
        (printout t ?taskID " Retract this workflow task because it needs step_file as input, which is not available according to the state model. "  crlf))
(defrule rule2
        (not (state_panel (available_resources $?x "gambit_jou_file" $?y)))    ?taskID<-(workflow_task(input $?a "gambit_jou_file" $?b))
=>      (retract ?taskID)
        (printout t ?taskID " Retract this workflow task because it needs gambit_jou_file as input,  which is not available according to the state model. "  crlf))
… …

(defrule workflow-answer-1
        (declare (salience -10))  (workflow_task (name ?n))
=>      (printout t "In term of the work flow, next step you can do: " ?n crlf))
```

As there are currently no semantically described EDSO task services available on the Grid, it makes no sense to search the Internet for any required task services. Therefore, we have not implemented the Semantics-based Search Engine of the service composition framework. In Geodise, the process of service discovery amounts to loading the EDSO task service ontology into the workflow construction environment. Users can then browse the service hierarchy and define appropriate services.

We have developed a workflow construction environment prototype for the framework as shown in Figure 4. The left panel is used to specify ontology services and the task service ontology. It presents the task service hierarchy through the Ontology Concept Browser. The right panel is the Component Editor. Its lower part is used to specify the properties of a task service and its upper part is used to search for task services that match the semantic description defined in the lower part. As we have not implemented the search mechanisms, the Component Editor is actually used to define a service directly. The middle panel is the Workflow Editor where services are composed and edited. The bottom panel is the State Monitor while the right top panel is used to display knowledge-based advice on service composition. The

knowledge-based advice system has not yet been wrapped up as a set of web services. It currently runs as a standalone knowledge-based system, which is directly integrated with the workflow construction environment. Despite this difference from the architectural specification detailed above, the decision support provided for service composition is the same.

A workflow specification represents a design solution to a specific EDSO problem. The general procedure for composing services as a workflow using the workflow construction environment is described step by step below. This process is also illustrated in Figure 4.

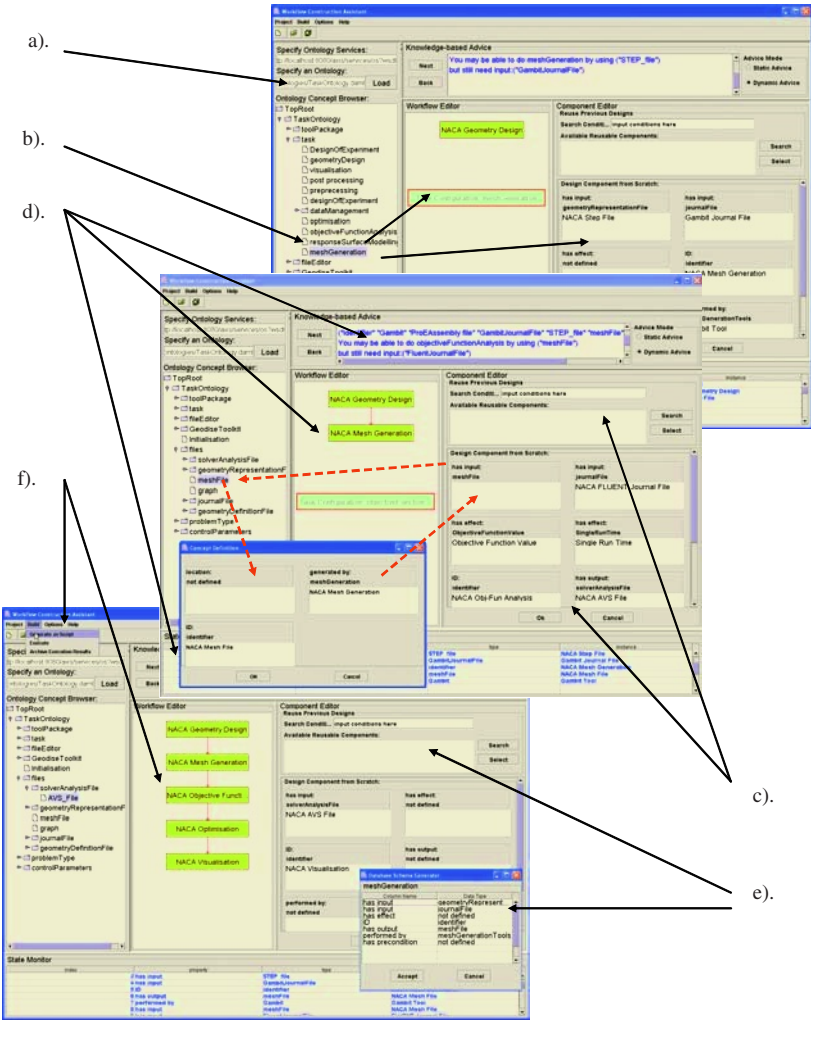

Fig. 4. Screenshots of Workflow Construction Environment

a). Specify and load the Geodise task service ontology via ontology services in the left panel. The Geodise task services will be presented in a hierarchy in the Ontology Concept Browser.

To start a workflow construction process, users need to provide an initial description of the problem at hand, e.g. the problem type and its characteristics. The knowledge-based advice system can then give advice on what to do first to solve the problem via the advice panel. Alternatively a static knowledge support system will suggest to users what should be done first.

b). Select a suitable primitive task service by navigating the service hierarchy utilizing the initial advice, and drag and drop it into the Workflow Editor. A task service description form will appear in the Component Editor for defining the service.

c). Define a task service by filling in the property values of the task service description form. Users can follow the ontological concept links from the semantic task service description to define each property. For example, to define a mesh file for the objective function analysis task, the semantic link of the property "meshFile" will bring you to the "MeshFile" concept in the Geodise task service ontology. Dragging and dropping the concept into the property's input area will in turn open a concept definition dialog box for users to input relevant values. This process is demonstrated by the red dashed arrows in Figure 4.

Alternatively users can partially describe the properties of a service using the service description form. The semantic-based search engine (at the top of the Component Editor) will enable users to discover similar task services on the Grid. This feature has not been implemented at present.

d). Once a task service is defined or discovered and selected in the Component Editor, two key operations will follow. First, an instantiated task service with embedded semantics will be added to the Workflow Editor. It will form a step of the workflow specified for the current problem. This is shown as a yellow box in Figure 4. Second some property information of the task service, in particular, the input, effect and output parameters, will be added to the state memory of the Workflow Construction Environment. These states are, in turn, passed on to the underlying advice system and displayed in the State Monitor. The recommendation on what one should/can do next is subsequently displayed in the knowledge advice panel. This advice guides users to select a suitable service from the service hierarchy.

e). A database schema for any task service can be generated automatically by dragging and dropping the service from the task service ontology. The instantiated service can then be archived in the database. By collecting all the services created for different problems a semantically-enriched knowledge base can be built over a period of time. This provides semantic content for the search engine to work on for future service discovery.

f). After an arbitrary number of loops, i.e. advice on required services, service discovery/configuration, and service composition, the user can construct a workflow that solves the specific problem. The generated workflow can be submitted to the underlying enactment engine where various resources will be bound together to form an executable. The executable will run in a domain specific execution environment. In Geodise, the executable is a Matlab .m script and the execution environment is the Matlab environment [32]. A full discussion of workflow enactment and execution issues is beyond the scope of this paper.

Each time a workflow is constructed for a particular design problem, it can be archived to form a semantically enriched problem/solution within a knowledge

repository. This facilitates the re-use of previous designs, while avoiding the overhead of manually annotating the solution with respect to semantic content.

6 Conclusions

This paper has described an approach, a framework and its implementation towards the delivery of knowledge-based service composition, or more generally, resource synthesis, in a web-enabled or Grid computing environment. A central feature of the approach discussed herein is the exploitation of domain-specific knowledge to compose web/Grid services into a workflow specification specifically geared to a core set of problem-solving objectives based on best practice knowledge and expertise. In developing this approach we have emphasized the importance of DAML-S, and related technologies, in providing semantically-enriched characterizations of available services as the basis for dynamic service discovery and appropriate resource utilization. We have further outlined a service-oriented architecture for knowledge-based systems operating in the context of the technological infrastructure provided by Grid-computing platforms and the semantic web. Our approach co-opts traditional knowledge-based systems engineering with the current state-of-the-art in ontology specification and XML web services. The prototype system, developed to provide a concrete demonstration of our approach, exemplifies this close merger of previously disparate technologies, availing itself of both a knowledge-based decision support facility and exploitation of semantically-enriched service descriptions in a single unitary environment. Such systems empower problem-solving agents by enabling maximal exploitation of available resources to meet a diverse set of complex problem-solving goals.

The approach and the example prototype have both been developed in a specific application context, namely that of design search and optimization. While the full evaluation of this system awaits further investigation and user feedback, our initial results have been promising. We have not seen any reasons to prevent this approach from being applied to other types of Grid applications.

The importance of domain knowledge and expertise to problem-solving success is nowhere more apparent than in the field of scientific computation and scientific discovery. We have demonstrated the importance of domain knowledge with respect to one aspect of expertise, namely the selection and configuration of services as part of a workflow specification. While this system is as yet only partially automated – users still need to manually construct workflows based on knowledge-system output – we firmly believe the current results are an important milestone on the way to providing a fully automatic means of intelligent service discovery and resource utilization in the context of Grid computing and the Semantic Web.

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) [33] and AKT (GR/N15764/01(P)).

References

1. Hey, T. and Trefethen, A. E.: The Data Deluge: An e-Science Perspective. To appear in "Grid Computing – Making the Global Infrastructure a Reality", Wiley, January 2003
2. Geodise project: http://www.geodise.org/
3. Foster, I. and Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)
4. Chinnici, R., Gudgin, M., Moreau, J. and Weerawarana, S.: Web Services Description Language (WSDL) 1.2, *W3C Working Draft*. http://www.w3.org/TR/wsdl12/ (2002)
5. Foster, I., Kesselman, C., Nick, J. and Tucke S.: The Physiology of the Grid: An Open Grid Service Architecture for Distributed Systems Integration. http://www.globus.org/ogsa/ (2002)
6. UDDI. The UDDI technical white paper. 2000. http://www.uddi.org/
7. SOAP 1.2 Working draft, http://www.w3c.org/TR/2001/WD-soap12-part0-20011217/
8. WSFL 1.0 http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
9. XLANG http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
10. Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., Thatte, S. and Weerawarana, S. Business Process Execution Language for Web Services, Version 1.0. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
11. IBROW project: http://www.swi.psy.uva.nl/projects/ibrow/home.html
12. Fensel, D., Benjamins, V.R., Motta, E. and Wielinga, B.: UPML: A Framework for knowledge system reuse. In Proceedings of the International Joint Conference on AI (IJCAI-99)
13. Benjamins, V.R., Plaza, E., Motta, E., Fensel, D., Studer, R., Wielinga, B., Schreiber, G. and Zdrahal, Z: IBROW3 – An Intelligent Brokering Service for Knowledge-Component Reuse on the World Wide Web. In proceedings of KAW'98 (1998)
14. Berners-Lee, T., Hendler, J. and Lassila, O.: The Semantic Web, Scientific American, May 2001
15. DAML+OIL http://www.daml.org.
16. DAML Services Ontology. DAML-:Web Service Description for the Semantic Web. In 1st International Semantic Web Conference (ISWC2001). http://www.daml.org/services/
17. OWL Web Ontology Language 1.0 Reference, http://www.w3.org/TR/owl-ref/
18. Sirin, E., Hendler, J. and Parsia, B.: Semi-automatic Composition of Web Services Using Semantic Descriptions. "Web Services: Modeling, Architecture and Infrastructure" workshop in conjunction with ICEIS2003 (2002)
19. Paolucci, M. Kawmura, T., Payne, T. and Sycara, K.: Semantic Matching of Web Services Capabilities. In The First International Semantic Web Conference (2002)
20. Extensible Markup Language, http://www.w3.org/XML/
21. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation Standard, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
22. Wroe, C., Stevens, R., Goble, C., Roberts, A. and Greenwood, M.: A **S**uite of DAML+OIL Ontologies to Describe Bioinformatics Web Services and Data. International Journal of Cooperative Information Systems, Editor(s): Prof. Bukhres, O. ISSN: 0218-8430 (2003)
23. Schreiber S., Akkermans H., Anjewierden A., Hoog R., and Shadbolt N.: Knowledge Engineering and Management. The MIT Press, London (1999)
24. Advanced Knowledge Technologies (AKT) project: http://www.aktors.org/
25. Cannataro, M. and Talia, D.: The Knowledge Grid. Communication of the ACM, Vol.46 No.1, pp89–93 (2003)
26. Chen, L., Cox, S.J., Goble, C., Keane, A.J, Roberts, A., Shadbolt, N.R., Smart, P. and Tao, F.: Knowledge Services for Distributed Service Integration. In Proceedings of UK e-Science all hands meeting (2002)

27. Tao, F., Chen, L., Shadbolt, N.R., Pound, G. and Cox, S.J.: Towards the Semantic Grid: Putting Knowledge to Work in Design Optimization. In Proceedings of the 3rd International Conference on Knowledge Management, Industry meets Science, pp555–566. (2003)
28. Jess, the rule engine for the Java platform, http://herzberg.ca.sandia.gov/jess/
29. Horrocks, I., Sattler, U. and Tobies, S.: Practical reasoning for expressive description logics. Lecture Notes in Artificial Intelligence, No.1705 pp.161–180. Springer-Verlag (1999)
30. Chen, L., Cox, S.J., Goble, C., Keane, A.J., Roberts, A., Shadbolt, N.R., Smart, P., Tao, F.: Engineering Knowledge for Engineering Grid Applications [http://www.geodise.org/VPO/Files/Papers/euroweb02Final-submission-version.doc]. In Proceedings of Euroweb 2002 Conference, The Web and the GRID: from e-science to e-business, pp12–25 (2002)
31. Bechhofer, S., Horrocks, I., Goble, C. and Stevens, R.: OilEd: a Reason-able Ontology Editor for the Semantic Web DL2001, 14th International Workshop on Description Logics (2001)
32. Pound, G., Eres, H., Wason, J., Jiao, Z., Keane, A.J., and Cox, S.J.: A Grid-enabled Problem Solving Environment (PSE) for Design Optimization within Matlab. International Parallel and distributed Processing Symposium IPDPS-2003 (2003)
33. MyGrid project: http://mygrid.man.ac.uk/index.shtml