Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Future Generation Computer Systems 20 (2004) 7–18

FGCS FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.com/locate/future

SGrid: a service-oriented model for the Semantic Grid

M. Li [a,*], P. van Santen [a], D.W. Walker [b], O.F. Rana [b], M.A. Baker [c]

[a] *Department of Electronic and Computer Engineering, Brunel University, Uxbridge UB8 3PH, Middlesex, UK*
[b] *Department of Computer Science, Cardiff University, PO Box 916, Cardiff CF24 3XF, UK*
[c] *Distributed Systems Group, University of Portsmouth, Portsmouth PO1 2EG, UK*

Abstract

This paper presents SGrid, a service-oriented model for the Semantic Grid. Each Grid service in SGrid is a Web service with certain domain knowledge. A Web services oriented wrapper generator has been implemented to automatically wrap legacy codes as Grid services exposed as Web services. Each wrapped Grid service is supplemented with domain ontology and registered with a Semantic Grid Service Ontology Repository using a Semantic Services Register. Using the wrapper generator, a finite element based computational fluid dynamics (CFDs) code has been wrapped as a Grid service, which can be published, discovered and reused in SGrid.

Keywords: Semantic Grid; Web services; Service-oriented model; Wrapper generator; Domain ontology

1. Introduction

A Grid [1] is a promising next generation distributed computing infrastructure, including the solution of large-scale resource intensive problems. The Grid couples a wide variety of geographically distributed resources such as PCs, workstations and clusters, storage systems, data sources, databases and special purpose scientific instruments and presents them as an unified integrated resource. A Grid instantiation is a Grid system prototype using one or more of the Grid middleware technologies such as Globus [2], Condor [3], or UNICORE [4]. Most current Grid instantiations are focused on computational services for end users. They lack the ability to provide domain problem solving services and knowledge related services.

Fundamental research on Semantic Web [5] has allowed the Grid community to move from the current data centric view supporting the Grid, towards a Semantic Grid with a set of domain specific problem solving services and knowledge services. The Semantic Grid [6] is a service-oriented architecture in which entities provide services to one another under various forms of contract. The Semantic Grid is characterized by an open system, with a high degree of automation, which supports flexible collaboration and computation on a global scale. In such an environment it is essential that information relating to the needs of the user and their applications, and the resource providers and their networking, storage and computational resources all have easily discoverable interfaces, and are defined, which means that can be used by higher-level services to effectively exploit the Grid. Zhuge [7] presents a soft-device concept to enhance semantic capabilities for the Grid by dynamically clustering resources through a single semantic image. Like the Semantic Web which is defined as “an extension of the current

* Corresponding author.
E-mail address: maozhen.li@brunel.ac.uk (M. Li).

0167-739X/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0167-739X(03)00160-2

Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation", the Semantic Grid can be described as an "extension of the current Grid in which information and services are given well-defined meaning, better enabling computers and people to work in cooperation" [8].

SGrid is a service-oriented model for the Semantic Grid in which each Grid service is exposed as a Web service (WS). A WS [9] is a self-contained, self-describing, modular application that can be published, located, and invoked across the Web. The core WSs include WSDL for service description, UDDI for service publishing and discovery, SOAP for service binding and invocation. Since the WSs standards are simple and based on standard Web technologies, these standards have the advantage of widespread academic and industrial support, which implies greater uptake, ease-of-use, and true ubiquity. In SGrid, each Grid service is exposed as a WS with a semantic description defined in a Semantic Grid Service Ontology Repository (SGSOR). A service consumer such as an end user needs to negotiate with a service provider about the terms under which the services can be provided. The underlying complexity of the Semantic Grid infrastructure and the speedy interaction required, make software agents the most likely candidate to handle these negotiations. In SGrid, each user accesses the Semantic Grid through an User Agent (UA). Each UA acts as a service consumer and interacts with a Grid System Agent (GSA), which acts as a service provider. The GSA manages all the Grid services in a Grid system. In addition, SGrid is end users oriented. End users are computational scientists knowing little about computer software and hardware. In SGrid, there are three situations for end users to access Semantic Grid services:

- End users provide a data set for a domain problem and a problem solver; SGrid only provides computational services. In this situation, an user domain problem solver can be wrapped as a Semantic Grid service to be used by other users.
- End users provide a data set and a description for a domain problem; SGrid provides information services (domain problem solvers) and computational services.
- End users provide a data set for a domain; SGrid provides knowledge services to find the semantic meaning of the data set.

The rest of this paper is organized as follows. Section 2 gives a layered structure for the Semantic Grid. Section 3 describes the software architecture of SGrid and the main components used in SGrid. Section 4 presents Web services oriented wrapper generator (WSOWG) for automatically wrapping legacy codes as Grid services exposed as WSs. Section 5 describes a case study on wrapping a computational fluid dynamic (CFD) code as a WS with WSOWG. Section 6 compares SGrid with related work. Section 7 concludes the paper and gives future work.

2. A layered structure for the Semantic Grid

From a service-oriented point of view, the Semantic Grid can be divided into four layers—computational services layer, data services layer, information services layer and knowledge services layer. The layered structure is shown in Fig. 1.

2.1. Computational services layer

The computational services layer is primarily concerned with large-scale pooling of computational resources. The services provided by this layer are related to resource discovery and allocation, resource monitoring, user authentication, task scheduling or co-scheduling, fault tolerance.

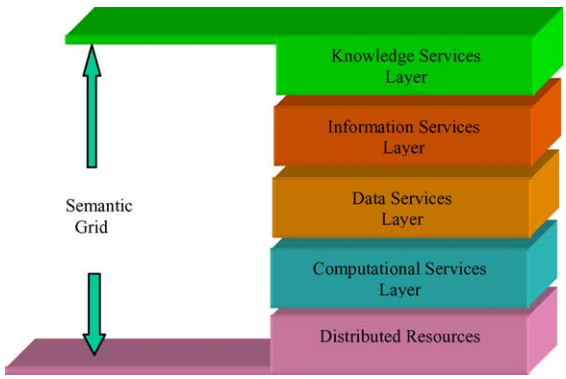

Fig. 1. A layered structure of the Semantic Grid.

2.2. Data services layer

Built on top of a computational services layer, a data services layer mainly provides intensive computation and analysis of shared large-scale data sets, from hundreds of TeraBytes to PetaBytes, across widely distributed scientific communities. The services provided by this layer are related to data storage, meta-data management, data replication and data transferring.

2.3. Information services layer

The information services layer runs on top of a data services layer allowing uniform access to heterogeneous information sources and providing commonly used services running on distributed computational resources. Uniform access to information sources relies on meta-data to describe information and to help integrate heterogeneous resources. The granularity of the offered services can vary, from subroutine or method calls to complete applications. Hence, in scientific computing, services can include the availability of specialized numerical solvers, such as matrix solvers and partial differential equation (PDE) solvers, to complete scientific codes for applications such as weather forecast and molecular or fluid dynamics. In commercial computing, services can be statistical routines based on existing libraries or prediction services, which offer coarser grained functionality, such as database profiling or visualization services. In hypermedia applications, services can be multimedia content analysis algorithms or hyperlink servers. Services in this layer can therefore be offered by individual providers or by corporations; they may be specialized for specific applications such as genomic databases, or general purpose, such as numerical libraries.

2.4. Knowledge services layer

The knowledge services layer is the top most layer that provides services, which can look for patterns in existing data repositories, and manage information services. Running on top of a data services layer, the knowledge service layer can provide knowledge discovery from a huge amount of data using a data-mining mechanism. Or it can provide semantic meaning of information services aggregated from the information services layer. This layer is domain oriented and usually uses domain knowledge built with domain ontology.

It is intended that each of these layers provide services to various applications, ranging from support for mobile devices, to large-scale single applications such as modeling protein folding and concurrent engineering.

A substantial part of the research effort dedicated to the Grid has concentrated on the computational and data layers. However, growing interest in the recently established “the Semantic Grid” working group at the Global Grid Forum [10] indicates the importance of services provided by the Semantic Grid.

3. The software architecture of SGrid

SGrid is a service-oriented model for the Semantic Grid. Fig. 2 shows the software architecture adopted by SGrid. The main components in SGrid are UAs and a GSA, a WSOWG used to wrap legacy codes as WSs, a Semantic Services Register (SSR) to register a WS with domain knowledge, a Semantic Services Mapper (SSM) to map from a semantic service description to a specific WS. The main functions of each component are given below.

3.1. Web services oriented wrapper generator

The WSOWG is used to automatically wrap legacy codes as WSs for use in SGrid. Each legacy code could be a problem solver, or a data-mining algorithm to discover a pattern in a data set. Each wrapped Grid service has a WSDL interface for service description; an entry in an UDDI based Grid Services Repository (UddiGSR) for service registry and discovery; a SOAP Listener for service implementation; a semantic capability description registered with a SGSOR.

3.2. Semantic Services Register

The SSR is used to extend the UddiGSR to support semantic services. The UDDI search mechanism relies on pre-defined categorization through keywords and does not refer to the semantic content of the advertisements. The registry is supposed to function in a

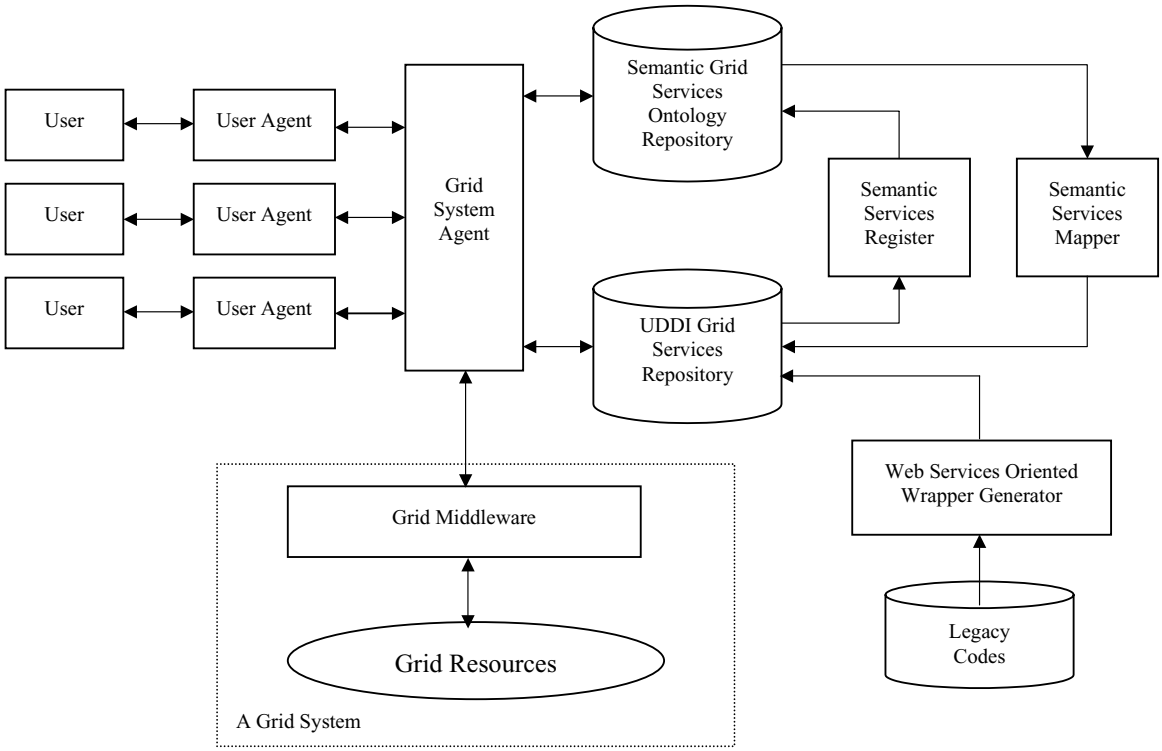

Fig. 2. The software architecture of SGrid.

```
<SemanticGridServices>
  <ServiceNode>
    <Name>Numerical Solvers</Name>
    <Parent>NULL</Parent>
    <Related query = "Numerical Libraries" match = "100%" />
    <Related query = "Numerical Calculation" match = "70%" />
    <Related query = "Numerical Analysis" match = "90%" />
  </ServiceNode>

  <ServiceNode>
    <Name>Partial Differential Equation</Name>
    <Parent>Numerical Solvers</Parent>
    <Related query = "PDE" match = "90%" />
    <Related query = "Partial Differential Solver" match = "100%" />
    <Related query = "Partial Differential Equation Library" match = "100%" />
    <Related query = "Method for Wave Propagation" match = "80% />
    <Related query = "Method for Heat Transfer in Solid" match = "80%" />
    <Related query = "Method for Flow in Porous Media" match = "80%" />
    <Related query = "Method for Plane stress and strain in structural Mechanics" />
    <Related query = "Mathematical Model" match = "40" />
  </ServiceNode>
</SemanticGridServices>
```

Fig. 3. The capability description of a Semantic Grid service.

```
for each entry in the SGSOR
    match the entry with a service request
    if match successfully
        find the leaf node related to the matched entry in the SGSOR
        add the leaf node with a match degree to a service candidate list
    continue match
end of match

for each entry in the service candidate list
    find the closest service candidate based on a match degree
return the closest service candidate registry
```

Fig. 4. The algorithm used to match a Semantic Grid service.

fashion similar to white or yellow pages, where businesses can be looked up by name or by standard service taxonomy as is already used within the industry. UDDI itself does not support semantic descriptions of services.

Each Grid service has a domain ontology description defined in XML to express its semantic capabilities. Ontology is defined as a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of interference and logic for some particular topics [11]. The SSR registers the semantic capability description of each Grid service with the SGSOR. Each Grid service in the UddiGSR has an unique registry in the SGSOR. The SGSOR is organized in a tree structure and defined in XML. Each registry in the SGSOR has a name and one or more related items of which each has a match degree to define the semantic relationship with the registry. For example, Fig. 3 shows the semantic capability description of a Grid service defining numerical solvers with a PDE service. The semantic capabilities for a PDE Grid service could be, for example, a PDE solver, a method for wave propagation, or a method for heat transfer in solid.

```
String findSemanticGridService (ServiceRequest)
{
    String Service;
    int i, j, NumberofMatch = 0;

    for ( i = 0; i < n; i++)     //n is the total entry number in SGSOR
    {
        if  (match (Entry_i,  ServiceRequest))
        {
            Service[NumberofMatch] = N_i, // find the leaf node N_i related to Entry_i;
            Match[NumberofMatch] = DegreeofMatch_i;
            NumberofMatch++;
            }
        }
     for ( j = 0; j <NumberofMatch ; j++)
    {
        if  (Match [0] < Match [j])
        {
            Match[0] = Match[j];
            Service = Service [j];
            }
        }
   return Service;
   }
```

Fig. 5. The implementation of the semantic service matching algorithm.

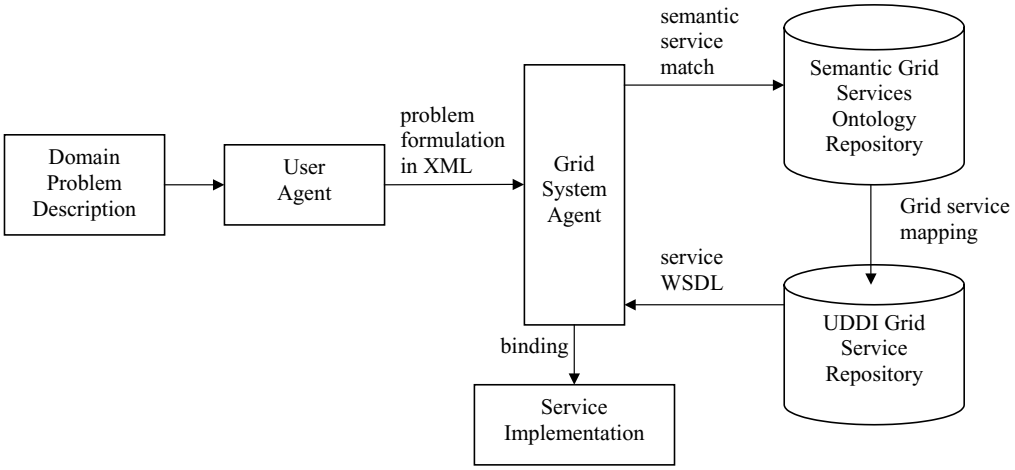

Fig. 6. The data flow for a Semantic Grid service matching.

3.3. Semantic services mapper

The SSM is used to match a specific WS in the UddiGSR from a semantic service registry in the SGSOR. Each registry in the SGSOR is a semantic service associated with a couple of related terms with different ranking rates. For an user service request, there may be multiple semantic service candidates based on a semantic understanding. The SSM will rank these semantic match degrees to choose the closest candidate. When searching and finding a Grid service, a semantic match will be performed first in the SGSOR, and then the WS in the UddiGSR can be found by mapping the registry in the SGSOR to the WS in the UddiGSR. The algorithm used to find a semantic service in the SGSOR is given in Fig. 4 and the implementation of the algorithm is given in Fig. 5.

3.4. UAs and GSA

The UA acts on behalf of a user. Each UA runs on a specific Grid system managed by a GSA. An UA communicates with the GSA to request Grid services available for an user. These Grid services are described in a user friendly way. For a molecular dynamic code, for instance, its functionality and usage information will be provided to an user. An user can browse Grid services available and then submit a service request.

An UA can assist an user in formulating a domain problem, and submit the user request to a GSA. For example, an user may ask for a service such as "*Find me a solution to partial differential equation*". The UA will then parse the user request, formulate it in XML and forward it to the GSA. The GSA will use the SSM to perform a semantic match in the SGSOR and map a specific WS in the UddiGSR, and then bind the service for service invocation. The data flow to find a Semantic Grid service is shown in Fig. 6.

4. The implementation of WSOWG

In this section, we will describe the implementation of WSOWG, a WSOWG used to automatically wrap legacy codes as WSs for reuse in the SGrid. First we give a brief introduction to a wrapper generator (WG), used to automatically wrap legacy codes as CORBA components.

4.1. A CORBA oriented WG

We have implemented WG, a CORBA oriented WG [12] on Solaris 2.7 using Visibroker [13] as the CORBA ORB. The main constraints for a legacy code to be wrapped as a component with WG are:

- The legacy code can be a sequential code or a parallel code using MPI [14].
- The legacy code can be written in C, Fortran or Java.

- The legacy code can be located anywhere within a distributed computing network.
- The legacy code must be a binary code and can perform certain functions with some input(s)/output(s).

When using the WG to wrap a legacy code, developers need to supply WG with parameters such as the wrapped component name, the name and location of the legacy code, the legacy code type (sequential or parallel in MPI), processors used, inputs and outputs of the legacy code. After receiving the parameters from a developer, WG will check the validity of these parameters, and then generate a CORBA IDL and an XML interface based on the input parameters. Using an IDL compiler, WG generates a skeleton for the component. Based on the specified parameters, WG knows how to generate the Listener. If there are no data input to the component, the generated Listener will automatically invoke the Body of the component once a request has been received. Otherwise, the generated Listener will first finish receiving data from another component, and then it invokes the Body of the component. Input data may either be streamed to the component, or read from a file. The Publisher is generated in a similar way to the Listener. The generation of the Body is completed by a Body template within WG, making use of the skeleton created by the IDL compiler. The main function of the Body is to invoke the legacy code wrapped inside it. After generating all the interfaces needed to wrap the legacy code as a component, WG stores the component in the Component Repository (CR) in XML for future use.

4.2. Web services oriented wrapper generator

In order to augment the WG to the WSOWG, we have implemented three new tools using Apache Axis [15] and UDDI4J [16] and integrated them with the WG. The functionality of each tool is described below.

4.2.1. IDL2WSDL

Based on Java2WSDL from Apache Axis, IDL2WSDL can generate a WSDL interface including the binding information for a component wrapped from a legacy code. IDL2WSDL makes use of an XML description of the component IDL generated by WG. CORBA parameters can be of type *out* or *inout*. An *out* type is a variable that is initialized and set in a server and sent back to the client. An *inout* type is a type that is initialized and set in a server and can be sent between a client and a server. IDL2WSDL deals with CORBA *out* and *inout* in this way. If an operation has *out* parameters then these parameters appear as parts in the operation response message. If an operation has *inout* parameters, then these parameters will have a part with the same in both the operation request and response messages. The WSDL interface can be used by a WS client to invoke the WS component, and can also be registered with the UddiGSR for service discovery.

4.2.2. WSDL2SOAP

WSDL2SOAP makes use of WSDL2Java from Apache Axis and the WSDL of a CORBA component to generate a SOAP Listener to implement a SOAP binding implementation for a WS component. The SOAP Listener listens requests from a WS client and then invokes the corresponding CORBA component wrapped via WG.

4.2.3. UDDIKit

A tool used to manage the UddiGSR. UDDIKit registers a WS component wrapped via WSOWG with the UddiGSR and publishes them on the Web. A Web user can find a service needed through browsing the UddiGSR.

Fig. 7 shows the data flow of WSOWG. After receiving the parameters for wrapping a legacy code as a WS component, WSOWG first invokes WG to check the validity of these parameters, such as the validity of the component directory, and whether the input types belong to the types it supports. WG will then generate a CORBA IDL and an XML interface based on the legacy code properties. Using an IDL compiler, WG generates a skeleton for the component including a Listener, a Publisher, and a Body of the component. Secondly, WSOWG invokes IDL2WSDL to generate a WSDL interface based on an XML description of the component IDL. Thirdly, WSOWG invokes WSDL2SOAP to take the WSDL interface as an input to generate a SOAP Listener, which will integrate a call to the CORBA component Listener through IIOP. Finally, WSOWG invokes the UDDIKit to register the wrapped component with the UddiGSR for service discovery.

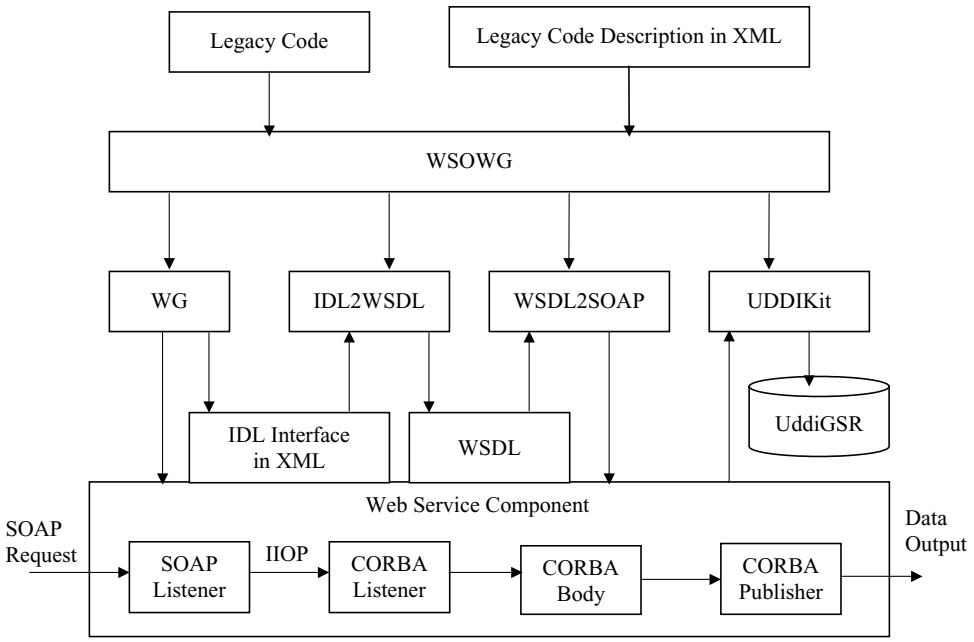

Fig. 7. The data flow of WSOWG.

5. A case study: wrapping a CFD code as a WS

In this section we describe how to use the WSOWG to automatically wrap a CFD legacy code as a WS component in SGrid. The CFD code called PHI3D [17] written in Fortran is a finite element based CFDs code for simulating incompressible Navier–Stokes flows, and is being used to model flow in the lung and upper respiratory system. The legacy code "PHI3D.x" makes use of one processor. It has 1 input and 12 outputs (steps, inner steps, Mass, Uvel, Vvel, Wvel, Temp, PHI, Press, DeltaT, Time, Reyn). The CORBA component IDL generated from the WG is illustrated in Fig. 8.

```
module CFD
{
    interface CFDComponent
    {
        void Listener(in string ComponentID, in string inputs);
        void Body(in string parameters);
        void Publisher(out string Component ID, out string outputs);
        };
    };
```

Fig. 8. The CORBA component IDL.

A segment of the WSDL interface describing the Publisher is given in Fig. 9. The WSDL interface is automatically generated through the IDL2WSDL tool provided by the WSOWG. Once the legacy code has been wrapped as a WS, the WSOWG will register the WS with the UddiGSR and invoke the SSR in SGrid to register its semantic capabilities with the SGSOR for semantic reuse.

6. Related work

We compare SGrid with related work based on the following aspects.

6.1. SGrid portals and legacy portals

Most Grid systems provide services to end users via a Grid portal. A Grid portal is a Web based user interface that provides seamless access to heterogeneous Grid resources. The goal is to allow scientists to focus completely on science by making the Grid a transparent extension of the user's desktop environment. Grid portals in current use include XCAT [18], Gateway [19], HotPage [20], JiPANG [21], AliEN [22], Grappa

```
<?xml version = "1.0" encoding = "ITF-8"?>
<definitions
    name = "CFDComponent"
    …
    <message name = "PublisherRequest" />
    <message name = "PublisherResponse" />
        <part name = "ComponentID" type = "xsd:string" />
        <part name = "outputs" type = "xsd:string" />
    </message>

    <portType name = "CFDComponent">
 …
        <operation name = "Publisher">
            <input message = "tns:PublisherRequest" />
            <output message = "tns:PublisherResponse" />
        </operation>
    </portType>

    <binding name = "CFDComponentBinding" type = "tns:CFDComponent">
        <soap:bindings style = "rpc" transport = http://schemas.xmlsoap.org/soap/http />
 …
        <operation name = "Publisher">
            <soap:operation soapAction = "Publisher" />
            <input>
                <soap:body use = "encoded" namespace = "urn:CFDComponent"
                 encodingStyle = http://schemas.xmlsoap.org/soap/encoding/ />
            </input>
            <output>…</output>
        </operation>
    </binding>

    <service name = "CFDComponentService">
        <port name = "CFDComponent" binding = "tns:CFDComponentBinding">
        <soap:address location = http://localhost:8080/axis/services/CFDComponent />
        </port>
    </service>
</definitions>
```

Fig. 9. The Publisher WSDL of the CFD WS component.

[23], Genius [24] and many others. While no two portal designs are the same, they all share the following characteristics [25]:

- A portal user makes a secure connection from a Web browser to the portal server.
- The portal server then obtains a certificate from a proxy certificate server and uses that to authenticate the user with the Grid.
- When the user completes defining the parameters of the computation the portal Web server launches an application manager, which is a process that controls and monitors the actual execution of the Grid computation. The Web server delegates the user's proxy credential to the application manager so that the application manager may act on the user's behalf.

In some systems, the application manager publishes an event/message stream to a persistent event channel archive. This event stream describes the state of the Grid application execution and can be monitored by the user through the Web browser.

Compared with a conventional Grid portal, the benefits provided by a portal built from SGrid can be summarized as follows.

6.1.1. Interoperability

The three-tier architecture results in a classic stove-pipe problem: user interfaces are locked into particular middle tiers, which in turn are locked into specific back end systems and resources. An SGrid portal built from Grid services, which are defined in a context-independent way in terms of their interfaces, rather than its implementations. Portals built this way are interoperable. An SGrid portal may use Grid services provided by different Grid systems allowing users to access coarse-grained federated Grid resources and services.

6.1.2. Semantic Grid services

An SGrid portal can provide information and knowledge services except for normal computational services provided by conventional portals. In addition, the use of software agents and domain ontology can semantically assist users in formulating their problem description, searching possible solutions on the Grid.

6.2. SGrid and the Semantic Web

SGrid is a service-oriented model for the Semantic Grid. It has a layered structure in which different layers provide different services. Research on the Semantic Web [26] can be applied to the knowledge services layer in SGrid. Zhuge [27] presents a knowledge Grid model and a knowledge Grid operation language (KGOL) to allow users to easily create and share their knowledge Grids. SGrid allows users to easily wrap legacy codes as Grid services including computational services, information services and knowledge services using the WSOWG. The KGOL can be used to express the semantic relationships of the wrapped Grid services in SGrid.

6.3. The WG in SGrid

There is some prior work that addresses issues for generating wrappers with semi-automatic generation or meta-wrapper style to leverage legacy codes to a distributed environment.

Vidal et al. [28] suggests wrappers and mediators to access data from heterogeneous database or legacy servers. Ashish and Knoblock [29] suggested that information mediator for obtaining information from multiple Web sources. Souder and Mancoridis [30] provided wrappers for securely integrating legacy systems into a distributed environment. However, these works are about how to migrate legacy information systems to a distributed environment mainly on obtaining information from multiple data sources. A wrapper is provided for each data source. WSOWG is about automatically leveraging legacy codes as WSs to a Grid environment.

Kim and Bieman [31] provided a wrapping technique that enables various legacy systems to be reused on CORBA based distributed environments without any changes to them. An automatic wrapper generation method based on extensible wrapping template classes is presented for wrapping legacy codes. The legacy codes in the work are sequential codes. WSOWG differs from the work in two ways. First, the legacy codes in WSOWG can be sequential codes or parallel codes using MPI based on the implementation of WG. Second, WSOWG is a software tool, not just a template class. Therefore, users do not need to write any codes to use WSOWG. They only need to specify the parameters related to a legacy code when wrapping the legacy code as a WS component with WSOWG.

SWIG [32] is a software tool that provides an interface compiler that connects high legacy codes written in C, C++, and Objective-C with scripting languages such as Perl, Python, and Tcl/Tk. Legacy codes can be sequential codes or parallel codes using MPI. Whereas SWIG focuses on manipulating legacy codes through the use of scripting languages, WSOWG is used to automatically wrap legacy codes as WS components (component SOAP Listeners, component WSDL interfaces, component CORBA skeletons, component data flow) which can then be plugged together to create applications for reuse on the Grid.

Built from the WG, the WSOWG also provides CORBA the ability for parallel computing. CORBA enables the seamless integration of distributed objects within one system, and is designed primarily for sequential applications. High performance computing applications are mostly parallel programs using message passing paradigms such as MPI. CORBA cannot replace the MPI communication layer due to architectural and performance constraints. When wrapping an MPI based high performance legacy code as a parallel CORBA component, WSOWG makes use of an MPI runtime to manage the intra-communication

of multiple processors within the parallel component. The inter-communication among different components is managed by CORBA. The advantage is that users can use existing CORBA implementations (such as Visibroker, Orbacus [33] and others) without any modification to CORBA IDL compilers, as is done in other projects with a similar objective, such as PARDIS [34] and Cobra [35].

7. Conclusions and future work

The Semantic Grid will play a very important role in the widespread uptake of the Grid. It will provide enhanced support for end users to access heterogeneous Grid services and resources by understanding their domain problems and providing solutions. SGrid is a service-oriented model for the Semantic Grid in which each Grid service is exposed as a WS with semantic capabilities defined in a domain ontology repository. Legacy codes can be automatically wrapped as Grid services exposed as WSs for semantic reuse in SGrid using the WSOWG. In addition, the use of software agents assists end users in formulating their domain problems, providing solutions, and submitting tasks to the Grid.

The SGrid project is at a very early stage. There are several key issues need to be resolved before its full implementation:

- SGrid should be able to support multiple GSAs. If a Grid service in one Grid system is not available, the GSA could negotiate with other GSAs to request the specific service. Thus SGrid could provide users the ability to access federated Grid resources and services, which may run across different domains.
- An expressive Semantic Grid Service Ontology should be taken into consideration using a semantic services description language such as DAML-S [36] to provide rules of inference and logic for some domains.

References

[1] I. Foster, C. Kesselman, The Grid, Blueprint for a New Computing Infrastructure, Morgan Kaufmann, San Francisco, USA, 1998.
[2] Globus. http://www.globus.org.
[3] Condor. http://www.cs.wisc.edu/condor/.
[4] UNICORE. http://www.unicore.de/.
[5] Semantic Web. http://www.w3.org/2001/sw/.
[6] Semantic Grid. http://www.semanticgrid.org.
[7] H. Zhuge, Clustering soft-devices in Semantic Grid, IEEE Comput. Sci. Eng. 4 (6) (2002) 60–62.
[8] N. Furmento, W. Lee, A. Mayer, S. Newhouse, J. Darlington, ICENI: an open Grid service architecture implemented with Jini, in: Proceedings of the IEEE/ACM SuperComputing'02, Baltimore, USA, November 2002.
[9] L.F.G. Sarmenta, Bayanihan computing net: Grid computing with XML Web services, in: Proceedings of the CCGrid 2002, Berlin, Germany, May 2002.
[10] Global Grid Forum. http://www.globalfridforum.org/.
[11] J. Hendler, Agents and the Semantic Web, IEEE Intell. Syst. 16 (2) (2001) 30–37.
[12] M. Li, O.F. Rana, M.S. Shield, D.W. Walker, A wrapper generator for wrapping high performance legacy codes as Java/CORBA components, in: Proceedings of the IEEE/ACM SuperComputing'00, Dallas, USA, November 2000.
[13] Borland. http://www.borland.com/visibroker.
[14] MPICH. http://www-unix.mcs.anl.gov/mpi/mpich/indexold.html.
[15] Apache Axis. http://www.xml.apache.org/axis/.
[16] UDDI. http://www-124.ibm.com/developerworks/oss/uddi4j/.
[17] P.T. Williams, A.J. Baker, Incompressible computational fluid dynamics and the continuity constraint method for the 3D Navier–Stokes equations, Num. Heat Transf. B 29 (1996) 137–273.
[18] S. Krishnan, et al., The XCAT science portal, in: Proceedings of the SuperComputing'01, 2001.
[19] M. Pierce, C.-H. Youn, G. Fox, The Gateway computational Web portal: developing Web services for high performance computing, in: Proceedings of the International Conference on Computational Science, 2002, pp. 503–512.
[20] M. Thomas, J. Boisseau, Development of Web toolkits for computational science portals: the NPACI HotPage, in: Proceedings of the HPDC9, 2000, pp. 308–309.
[21] T. Suzumura, et al., JiPANG: a Jini-based computing portal system, in: Proceedings of the SuperComputing, 2001.
[22] AliEN. http://www.alien.cern.ch/.
[23] Grappa. http://www.iuatlas.physics.indiana.edu/grappa/.
[24] Genius. http://www.genius.ct.infn.it/.
[25] D. Gannon, et al., Programming the Grid: distributed software components, P2P and Grid Web services for scientific applications, Cluster Comput. 5 (3) (2002) 325–336.
[26] H. Zhuge, VEGA-KG: a way to the knowledge web, in: Proceedings of the 11th International World Wide Web Conference, Honolulu, Hawaii, USA, May 2002.
[27] H. Zhuge, A knowledge Grid model and platform for global knowledge sharing, Expert Syst. Appl. 22 (2002) 313–320.
[28] M.E. Vidal, L. Raschid, J.R. Gruser, A meta-wrapper for scaling up to multiple autonomous distributed information sources, in: Proceedings of the Third International Conference on Cooperative Information Systems, 1998, pp. 148–157.
[29] N. Ashish, C.A. Knoblock, Semi-automatic wrapper generation for Internet information sources, in: Proceedings of the Second International Conference on Cooperative Information System, 1997, pp. 160–169.

[30] T. Souder, S. Mancoridis, A tool for securely integrating legacy systems into a distributed environment, in: Proceedings of the Sixth Working Conference on Reverse Engineering, 1999, pp. 47–55.
[31] H.S. Kim, J. Bieman, Migrating legacy systems to CORBA based distributed environments through an automatic wrapper generation technique, in: Proceedings of the Joint Meeting of the Fourth World Multi-Conference on Systematics, Cybernetics and Informatics (SCI'2000) and the Sixth International Conference on Information Systems Analysis and Synthesis, ISAS'2000.
[32] D.M. Beazley, P.S. Lomdahl, Lightweight computational steering of very large scale molecular dynamics simulations, in: Proceedings of the IEEE/ACM SuperComputing'96, 1996.
[33] Orbacus. http://www.ooc.com/ob.
[34] K. Keahey, D. Gannon, PARDIS: a parallel approach to CORBA, in: Proceedings of the Sixth IEEE International Symposium of High Performance Distributed Computation, 1997, pp. 31–39.
[35] T. Priol, C. René, Cobra: a CORBA-compliant programming environment for high-performance computing, in: Proceedings of the Euro-Par'98, 1998, pp. 1114–1122.
[36] M. Paolucci, T. Kawamura, T.R. Payne, K. Sycara, Semantic matching of Web services capabilities, in: Proceedings of the First International Semantic Web Conference (ISWC), June 2002.

M. Li is a Lecturer in Distributed Systems at Brunel University. He received the PhD from Institute of Software, Chinese Academy of Sciences in 1997. He was a Research Associate in Department of Computer Science, Cardiff University from January 1999 to January 2002. Dr. Li's research interests are in the areas of Semantic Grid, parallel and distributed computing, information retrieval, multi-agent systems, multi-modal user interface, computer supported for cooperative work.

P. van Santen leads the Distributed and Grid Computing Group at the Department of Electronics and Computer Engineering at Brunel University, and serves as the chief Architect for the new High Performance cluster facility in the Brunel Information Technology Laboratory (BITlab). He is a principal investigator for a project concerned with Service Discovery and Interoperability funded by the EU-DataTAG project through PPARC. His present research interests lie in the areas of parallel system performance analysis, high performance clusters and Grid computing. He holds a lectureship in Advanced Computer Architecture and Distributed Computing.

D.W. Walker is head of the Parallel and Scientific Computing Group in the Department of Computer Science at Cardiff University, and the Director of the Welsh E-Science Centre. Professor Walker's research interests focus on software, algorithms, and environments for computational science on high performance computers. He has been closely involved in the development of the ScaLAPACK parallel software library, and the MPI message passing standard. He has also contributed to the design of a parallel version of the Community Climate Model, and has published a number of papers on the parallel implementation of particle-in-cell algorithms for plasma simulations. He has also been involved in the benchmarking of science and engineering applications codes on parallel computers. Prof. Walker has published over 70 papers in the area of parallel computing and has co-authored three books on the subject.

O.F. Rana is a Senior Lecturer in Computer Science at Cardiff University, and the Deputy Director of the Welsh E-Science/Grid Computing Centre. He also acts as advisor to Grid Technology Partners —an US based company specializing in Grid technology transfer to industry. He holds a PhD in Computer Science from Imperial College, London University in parallel architectures and neural algorithms, MSc in Micro-electronics from Southampton University, and BEng in Information Systems Engineering from Imperial College, London. His research interests are in the areas of high performance distributed computing, multi-agent systems and data-mining. Prior to joining Cardiff University, he worked for over 5 years in the use of neural algorithms in a number of fields, including biotechnology, instrumentation and control.

M.A. Baker is a Reader in Distributed Systems within the Faculty of Technology at the University of Portsmouth. Dr. Baker leads the Distributed Systems Group (DSG) that is actively working in the areas of distributed Java, Cluster Computing, and Grid Technologies. Dr. Baker is co-founder and co-chair of the IEEE Computer Society's Task Force on Cluster Computing (TFCC). Dr. Baker is an editor of the international journal *Computation and Concurrency*: *Practice and Experience* (Wiley) and of IEEE Computer Society's digital magazine (DS-online).