Developing Services for Design Optimisation on the Grid

Xue, G, Song, W, Keane, AJ and Cox, SJ
{gx, w.song, ajk, sjc}@soton.ac.uk
School of Engineering Sciences
University of Southampton
SO17 1BJ, UK

Abstract

In this paper we present our work on implementing elemental services for carrying out design optimisation on the Grid. The service-oriented approach makes it possible for to harness best available technologies that are usually incompatible due to heterogeneous software environments. It also facilitates the creation of integrated Problem Solving Environment (PSE) for design optimisation. We explain in details how state-of-the-art Web services technologies are exploited to build these services and yield desired quality of service, and demonstrate how service based design optimisation workflows constructed using script languages are used to solve design optimisation problems from different paradigms.

1. Introduction

Computer based design optimisation processes are broadly employed in many scientific and engineering areas to solve complicated problems such as computational fluid dynamics (CFD) and finite element methods (FEM), so as to produce the optimal results for the designers. Bringing improvements to the design optimisation processes to make them more accurate and efficient can provide the cutting edge in research and industrial competitions. So far, there have been many results from efforts to enhance the design optimisation processes made in the following two directions: (i) to bring in more computation resources for better quality and higher performance; and (ii) to develop better design optimisation methods for more accurate design modelling and more efficient optimisation search. It has, however, become a daunting task for designers to take full advantage of all these achievements in their daily design activities, as it requires integration of various elements that are either developed using incompatible software technologies, or deployed in heterogeneous environments.

The development and maturing of Grid computing technologies has provided a new solution to tackle the integration problem in design optimisation, as one of the main target of the Grid technologies is to enable large-scale, dynamic collaborations among participants from highly heterogeneous and distributed environments. The application of Grid technologies has facilitated the research and development of dedicated problem solving environments for design optimisation, such as the Geodise system [1].

With the introduction of OGSA [2] and OGSI [3], as well as the following WS-Resource framework [4], service-oriented Grid computing has been widely accepted by the Grid community. The design optimisation processes built in Geodise is fully compliable with the service-oriented approach by using services to assist designer at all stages of the design process. All technologies applied in the processes, regardless of what programming language they are developed in or what the platform they run on, are encapsulated into standard Web/Grid services that are universally accessible. It is therefore possible to achieve seamless integration of these heterogeneous technologies. In addition, this also decouples design optimisation components from each other, and therefore makes the system more flexible and extensible.

The service-oriented architecture itself, however, can not meet all demands for building design optimisation processes. The processes often include many sophisticated operations that require careful management of application state, guaranteed and faultless message delivery, and tightly controlled security. In implementing our design optimisation services, a number of advanced Web/Grid service technologies have been exploited to address these issues.

The rest of this paper is organised as follows. In the next section we describe the services developed for important steps in the design optimisation processes, and explain the advantage of using service-orientation. Section 3 provides details of the service implementation, focusing mainly on the advanced Web/Grid service features introduced to the development. To illustrate the use of the services, sample operations from different stages of the design optimisation process are demonstrated in Section 4. We draw the conclusions in Section 5.

2. Services for Design Optimisation

Figure 1 shows the workflow of our service-oriented design optimisation processes. Essential functions in

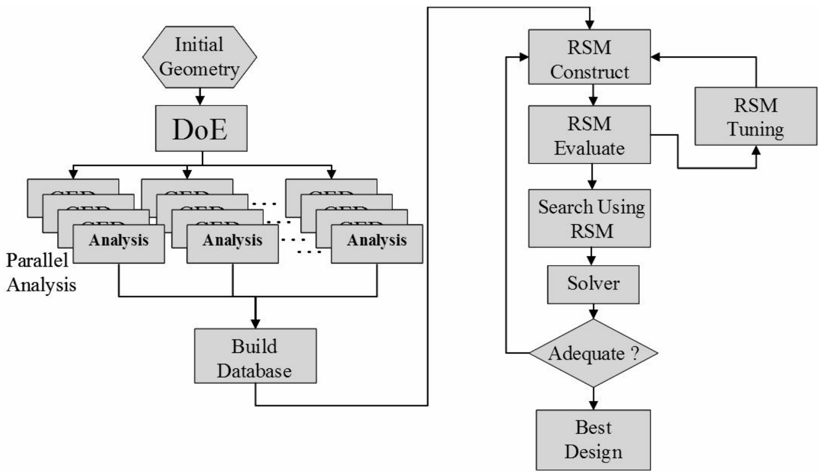


Figure 1: The Design Optimisation Process Workflow

the process, including design of experiment, response surface modelling and optimisation search have all been implemented as services, so that it is feasible to apply alternative design optimisation technologies to get the best results.

2.1. Service for Design of Experiment

The design of experiment (DoE) service provides a range of experiment design methods that are used to produce starting points required for generation of initial design geometries. To get the designs, the users need to specify ranges of the design parameter values, as well as the number of designs needed.

The DoE service is designed as a stateful service, which is able to store all designs generated based on a particular setting. Service users are therefore allowed to come back and require more designs based on the same settings without triggering another run of the applied DoE function. The usage of the service is therefore made more flexible and efficient.

The main DoE methods applied in the current service implementation are supplied by the OPTIONS system [5], including Latin Hypercube, Random, LP τ , Central Composite that are capable of meeting different design requirements.

2.2. Service for Response Surface Modelling

The response surface modelling (RSM) service implements various curve-fitting and regression methods available from OPTIONS, including Radial basis function, stochastic process modelling, and polynomial regression models. Results from the DoE service is passed onto response surface modelling service to build response surface which can be then tuned using additional design points when they become available.

Unlike the optimisation services, the RSM service is constructed as a normal service, as no internal state is required for the modelling operations. However, in order to support incremental response surface building and tuning, the service has introduced mechanisms to set up and manage operation context for a complete RSM process.

2.3. Service for Optimisation Search

The optimisation services expose numerical optimisation algorithms through reverse communication interfaces [7], which facilitate the decoupling of the optimisation system that drives the search process from the modelling codes that evaluate objective functions and constraints. It therefore becomes possible for the best optimisation methods to be deployed for a variety of design problems without any compatibility issues.

Strong support for state management is employed in the optimisation services, as for reverse communication based optimisations, the optimisers need to maintain information such as the control parameter values, the design parameter boundaries, and the search history throughout the entire process. In addition, considering that an optimisation operation usually involves a large number of search steps, mechanisms to ensure secured, reliable interactions between the client and the service have also been introduced.

Our optimisation services are constructed based on a framework which provides a generic abstraction of numerical optimisation operations. It can therefore be used to host diverse optimisation methods that can be called through reverse communication interfaces. So far we have built our optimisation services around the OPTIONS genetic algorithms, as well as the open source PORT library. The services have been successfully applied to optimise different design problems, such as those demonstrated in section 4, as well as in [6] and [7].

Apart from the design optimisation services described above, a number of general purpose services, such as the computation service [8] and the data service [9], have also been deployed to assist resource-intensive operations in the process.

3. Implementation

Basic Web services technologies, including XML Schema, SOAP and WSDL, have established a simple but sound infrastructure for the exchange of XML documents that are successfully used in our services to provide access to design optimisation functionalities. Yet operations for design optimisation are often complicated, involving multiple stages and numerous interactions with the services. It demands support for stateful service interactions and mechanisms that ensure right sequences of execution. In addition, security has remained as a major concern. To address these requirements, our service implementations have incorporated a number of enhancements to the basic Web services framework.

Apart from the efforts on the service layer, our implementations also include work on the integration of legacy design optimisation technologies, and the development of service clients that facilitate composition of services in design environments.

3.1. Enabling Stateful Service Interactions

The basic Web services specifications do not bear any notion of application state, which is nevertheless essential to many design optimisation operations such as response surface modelling and optimisation search. In general, two different approaches are applicable to solve this problem: the transient grid service model and the use of operation context.

The transient grid service model is first proposed together with OGSA and then technically specified in OGSI. The basic idea is to create 'instances' of the service that are responsible for maintaining all operation-specific state. Similar to distributed objects, the service instances are instantiated through a factory service and are identified using the Grid Service Handlers (GSH) [3]. All service interactions for a particular operation are carried out toward the same service instance. The instances only exist for the lifetime of the operations and are destroyed once the operations are finished. The use of transient grid services is illustrated in Figure 2.

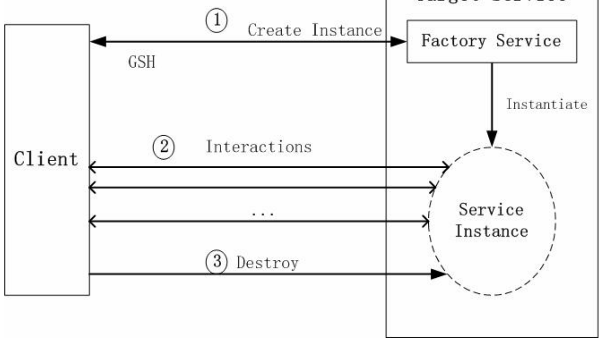
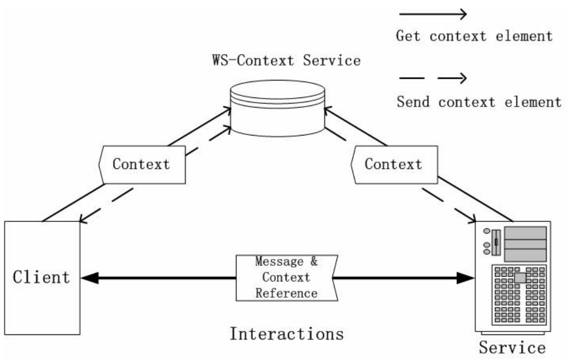


Figure 2: The OGSI Transient Service Model

Instead of building an external infrastructure like OGSI, the WS-Context [10] specification attempts to provide support for stateful service interactions within the Web services framework. It adheres to the principle that Web services are stateless, and tries to encapsulate operation state in an XML based entity called 'context'. Each interaction message for a particular operation contains in the SOAP header its context, from which the target service can retrieve state information so as to restore the right conditions and settings. A context can be created by the target service, or by a WS-Context Service designed in the same specification. The context may be augmented or modified each time the service is invoked. The WS-Context Service is also used to maintain the context entities. For operations with big amounts of state information, the service messages can only provide a URI reference to contexts stored on the service. Figure 3 illustrates how context works, together with a sample context entity in the SOAP header.



(a) Context based Stateful Interactions

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2002/06/soap-envelope">
   <soap:Header encodingStyle=
      "http://www.webservicestransactions.org/schemas/wsctx/2003/03"
      mustUnderstand="true">
      ...
      <context xmlns=
         "http://www.webservicestransactions.org/schemas/wsctx/2003/03"
         timeout="100">
         <context-identifier>
            http://aries.sesnet.soton.ac.uk/RSMService/
            6d05a24f-8132-494a-8a18-1171c1634f79
         </context-identifier>
         <activity-service>
            http://draco.sesnet.soton.ac.uk/CtxService/Manager.asmx
         </activity-service>
         <activity-list>
            <service>
               http://aries.sesnet.soton.ac.uk/RSMService/Service.asmx
            </service>
         </activity-list>
      </context>
      ...
   </soap:Header>
   <soap:Body>
      <!-- Application Payload -->
   </soap:Body>
</soap:Envelope>
```

(b) A Sample Context Element

Figure 3: Using Context for Stateful Interactions

Both approaches described have been exploited in our implementations to best suite the conditions of individual services. For instance:

The OGSI transient grid service model has been used in the implementation of the numerical optimisation service, which needs to make quick response to service invocations. An optimisation operation may involve hundreds or thousands of search steps. It is therefore not hard to see that for the optimisation service, the transient service approach is far superior in performance, since the service instance keeps all state information which can be used directly each time it is invoked. On the contrary, state of the optimisation search needs to be loaded every time if the context based approach is applied. Considering the large number of service interactions, the aggregate impact on performance is too significant to ignore.

Design optimisation services implemented in the transient grid service model have bee deployed using OGSI.NET [11] and MS.NETGrid [12]. Both platforms use the .NET framework [13] to create hosting environments for transient grid services and to provide comprehensive programming support for grid service development.

The context based approach for stateful service interactions is more suitable for services that are less critical in response time, and have large elements of state information. A typical example is the RSM service, which does not have frequent service interactions, but needs all submitted objective function values for each modelling action. Using WS-Context, the state information is only loaded when the service invoked, rather than staying in the system all the time. It reduces the cost of resource for running the service, and makes it more scalable.

Apart from performance considerations, the context based approach is more compatible with the service-oriented architecture defined in [14] and [15]. It puts fewer burdens on the service hosts, not requiring additional infrastructures such as OGSI. The deployment of design optimisation services is made simpler and easier.

Services using WS-Context have been deployed over normal Web service platforms such as ASP.NET. To handle the context element in the SOAP header, a message filter is added to the WSE [16] pipeline, which intercepts the context element and transform it to a programmable format. A context service partially implementing the WS-Context service has also been constructed.

3.2. Enabling Reliable Service Interactions

Service-oriented optimisation search usually involves numerous, repeated exchanges of similar messages between the client and the optimisation service. To carry out successful search operations, it is important that these messages are sent and received in the right sequence, and exactly once. However, in a heterogeneous environment, many errors such as system malfunction and network failure may occur to cause loss or duplication of messages, which can impair the search efficiency and even make the search fail.

In order to avoid potential errors caused by failed or false message delivery, features from the WS-ReliableMessaging [17] protocol have been implemented in the optimisation service. The protocol defines an XML element named 'sequence' to track and manage message deliveries. All messages of a particular operation should carry a *sequence* element, which contains a unique sequence identifier assigned to the operation, as well as an incremental message number. The sequence identifier and the message number combine to uniquely identify a message, while the message numbers specify the right order of delivery. Therefore, it is possible for the message receiver to detect delivery errors with a wrong sequence identifier or a non-incremental message number. Based on the establishment of mutual understanding of the *sequence*, WS-ReliableMessaging also defines mechanisms for message senders and receivers to confirm successful message deliveries, or report errors.

For our optimisation service, a *sequence* is established when a new service instance is created. Each request message to the service includes a sequence element in the SOAP header, and a request for acknowledgement of message delivery. In the response message, the service adds in information about all received messages, which can be used by the client to judge whether the response is rightly delivered corresponding to the request.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    ...
    <wsrm:CreateSequence
      xmlns:wsrm="http://http://schemas.xmlsoap.org/ws/2004/03/rm"/>
    ...
  </soap:Header>
  <soap:Body wsu:Id="Id-7946865e-472c-4b9d-b2cc-2b7df53e6ead"
      xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
    <createService xmlns="http://ogsa.gridforum.org/factory">
      <creation xmlns=
            "http://www.gridforum.org/namespaces/2002/10/gridServices">
        <serviceParameters
            xmlns="http://www.gridforum.org/namespaces/2003/03/OGSI">
            ...
        </serviceParameters>
      </creation>
    </createService>
  </soap:Body>
</soap:Envelope>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    ...
    <wsrm:CreateSequenceResponse
      xmlns:wsrm="http://http://schemas.xmlsoap.org/ws/2004/03/rm">
      <wsrm:Identifier>
        http://aries.soton.ac.uk/Optimisation/7ab61eec-ebae
        -430e-a2bb-35090b86f45c
      </wsrm:Identifier>
    </wsrm:CreateSequenceResponse>
    ...
  </soap:Header>
  <soap:Body wsu:Id="ee1db94c-d366-43e7-b9a1-a4c1d4f43c74"
      xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
    <createServiceResponse xmlns="http://ogsa.gridforum.org/factory">
      ...
  </soap:Body>
</soap:Envelope>
```

Figure 4: Creating Sequence on Service Instantiation

For example, in Figure 5, the client can conclude that the response message is wrongly delivered, as the range of received messages acknowledged does not match the message number of the request. The client then arranges a re-transmission to get the right response. To avoid confusing the service, a *Retransmission* element is added to differentiate from a duplicate message delivery.

```
<?xml version="1.0" encoding="UTF-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header
      xmlns:wsrm="http://http://schemas.xmlsoap.org/ws/2004/03/rm">
    ...
    <wsrm:Sequence>
      <wsu:Identifier>http://aries.soton.ac.uk/Optimisation/
            7ab61eec-ebae-430e-a2bb-35090b86f45c</wsu:Identifier>
      <wsrm:MessageNumber>10</wsrm:MessageNumber>
    </wsrm:Sequence>
    <wsrm:AckRequested>
      <wsu:Identifier>http://aries.soton.ac.uk/Optimisation/
            7ab61eec-ebae-430e-a2bb-35090b86f45c</wsu:Identifier>
    </wsrm:AckRequested>
    ...
  </soap:Header>
  <soap:Body>
    <Optimise xmlns="http://www.geodise.org/optimisation">
      ...
    </Optimise>
  </soap:Body>
</soap:Envelope>                                  Request Message
```

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header
      xmlns:wsrm="http://http://schemas.xmlsoap.org/ws/2004/03/rm">
      ...
    <wsrm:SequenceAcknowledgement>
      <wsu:Identifier>http://aries.soton.ac.uk/Optimisation/
            7ab61eec-ebae-430e-a2bb-35090b86f45c</wsu:Identifier>
      <wsrm:AcknowledgementRange Upper="9" Lower="1"/>
    </wsrm:SequenceAcknowledgement>
      ...
  </soap:Header>
  <soap:Body>
    <OptimiseResponse xmlns="http://www.geodise.org/optimisation">
      <OptimiseResult>
      ...
      </OptimiseResult>
    </OptimiseResponse>
  </soap:Body>                                False Response Message
</soap:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header
      xmlns:wsrm="http://http://schemas.xmlsoap.org/ws/2004/03/rm">
    ...
    <wsrm:Sequence>
      <wsu:Identifier>http://aries.soton.ac.uk/Optimisation/
            7ab61eec-ebae-430e-a2bb-35090b86f45c</wsu:Identifier>
      <wsrm:MessageNumber>10</wsrm:MessageNumber>
      <gd:Retransmission
            xmlns:gd="http://www.geodise.org/optimisation"/>
    </wsrm:Sequence>
    <wsrm:AckRequested>
      <wsu:Identifier>http://aries.soton.ac.uk/Optimisation/
            7ab61eec-ebae-430e-a2bb-35090b86f45c</wsu:Identifier>
    </wsrm:AckRequested>
    ...
  </soap:Header>
  <soap:Body>
    <Optimise xmlns="http://www.geodise.org/optimisation">
      ...
    </Optimise>
  </soap:Body>
</soap:Envelope>                             Message Re-Transmission
```

Figure 5: Use of WS-Reliable Messaging

In addition to ensuring message delivery, the sequence mechanism can also used to trace the optimisation search route. Since the message numbers are unique in the sequence, they are used to index the search history kept at the service. Using such indexes, users can easy request the restart of optimisation at a specified point.

Like the context element, sequence and acknowledgement elements are all processed by a customised WSE filter. Since it is not feasible to use customised filter under OGSI.NET, the implementation is so far only available on the MS.NETGrid platform.

3.3. Enforcing Security

Our design optimisation services follow the WS-Security [18] specification to control access to the services, as well as to protect message integrity and privacy. The implementations select to use X.509 [20] certificates for signing and encrypting SOAP messages.

In order to publish the security requirements, our design optimisation services make use of the WS-SecurityPolicy [20] specification, which identifies a number of WS-Policy [21] assertions for security issues. These assertions allow the service to state what security tokens are acceptable and what are the requirements for message integrity and confidentiality.

```
<wsse:SecurityToken wsp:Usage="required"
         xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
         xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy" >
  <wsse:TokenType>wsse:X509v3</wsse:TokenType>
  <wsse:TokenIssuer>E-Science CA</wsse:TokenIssuer>
</wsse:SecurityToken>
```

```
<wsse:Integrity wsp:Usage="wsp:Required"
         xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
         xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
         xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
  <wsse:Algorithm  Type="wsse:AlgCanonicalization"
         URI="http://www.w3.org/Signature/Drafts/xml-exc-c14n"/>
  <wsse:Algorithm Type="wsse:AlgSignature"
         URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <wsse:SecurityToken>
    <wsse:TokenType>wsse:X509v3</wsse:TokenType>
  </wsse:SecurityToken>
  <MessageParts
         Dialect="http://schemas.xmlsoap.org/2002/12/wsse#soap">
    soap:Body
  </MessageParts>
</wsse:Integrity>
```

```
<wsse:Confidentiality wsp:Usage="wsp:Required"
         xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
         xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
         xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
  <wsse:Algorithm Type="wsse:AlgEncryption"
         URI="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
  <wsse:keyinfo>
    <wsse:SecurityToken>
      <wsse:TokenType>wsse:X509v3</wsse:TokenType>
    </wsse:SecurityToken>
  </wsse:keyinfo>
  <MessageParts>
    wsp:GetInfosetForNode(wsp:GetBody(.))
  </MessageParts>
</wsse:Confidentiality>
```

Figure 6: WS-SecurityPolicy Assertions for the Design Optimisation Services

Figure 6 shows typical examples of WS-SecurityPolicy assertions used by the design optimisation services. They specify that an X.509 v3 certificate issued by the UK e-Science CA is required as the security token, that the services only accept messages which have the SOAP body element signed using *Exclusive Canonicalisation* and the *RSA-SHA1* algorithms with an X.509 certificate, and that the SOAP body element must be encrypted using the *RSAES-PKCS1-v1_5* algorithm.

Security features for the design optimisation services have been implemented with supports from the WSE package. Depending on the underlying service hosting environment, the security policies are created differently. On OGSI.NET, they are constructed programmatically using customised .NET attributes [22]. For services built on ASP.NET and MS.NETGrid, policy documents need to be created explicitly, and mapped to the services in the service configuration, as illustrated by Figure 7.

```
<configuration>
  <microsoft.web.services>
    ...
    <policy>
      <receive><cache name="securitypolicies.xml"/></receive>
      <send><cache name="securitypolicies.xml"/></send>
    </policy>
    ...
  </microsoft.web.services>
</configuration>
```

Figure 7: Mapping Security Policies to Services

3.4. Integrating Legacy Technologies

An important issue in building the design optimisation services is how to integrate design optimisation programs developed in the earlier stage with new software technologies used for service development. Most of these legacy technologies are only available either as source code in native programming languages such as FORTRAN and C that are much different from those used to build Grid/Web services, or as native binary libraries. There are generally two approaches that can be used to link the service programming environment and the legacy technologies, as shown in Figure 8.

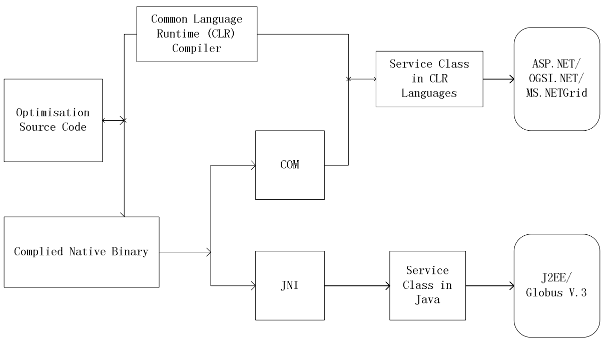

Common Language Runtime (CLR) Compiler
Optimisation Source Code
Service Class in CLR Languages
ASP.NET/ OGSI.NET/ MS.NETGrid
COM
Complied Native Binary
JNI
Service Class in Java
J2EE/ Globus V.3

Figure 8: Integrating Service with Legacy Design Optimisation Technologies

The first approach to breach the division between existing native source code and new programming languages makes use of the Common Language Runtime (CLR) [23], which provide multi-language support and seamless integration of code written in various languages, such as FORTRAN, C/C++ and C#. It is therefore possible to have existing source code compiled into CLR libraries, and accessed directly by service also developed in CLR environment. This approach has been applied in the implementation of the optimisation service for grading based optimisation algorithms from the open-source PORT library [27], as well as the computation service for interactions with local resource management systems [8].

The CLR approach requires direct access to source code of the legacy technologies. In many cases, however, they are only available as compiled native binary libraries. The second approach addresses this problem by building native interfaces to the binaries using technologies such as Java Native Interface (JNI) [25] and Common Object Model (COM) [27], so that the native binaries can also run under the selected platforms. If JNI is involved, the services need to be built on Java based Grid/Web services environments, while the COM based approach allows CLR based service environments to be used. This approach has been applied in the implementation of optimisation service for genetic algorithms from OPTIONS, as well as the RSM service and the DoE service.

3.5. Building Service Based Workflow

We choose to apply script languages such as Matlab and Jython for the construction of the service based workflow for design optimisation operations, instead of Web services orchestration technologies such as BPEL4WS [27]. Compared to business processes, design optimisation operations are often more complicated, which requires the flexibility of scripting languages. In addition, scientists and engineers carrying out design optimisation are usually more familiar with scripting languages and design environments such as Matlab.

In order to facilitate service interactions from the scripting environments, clients for the design optimisation services have all been constructed in Java, which connects to the scripting languages seamlessly as the Java virtual machine has been integrated into environments including Jython and Matlab. The service clients follow the dynamic, filter based structure described in [8] for flexibility and adaptability. Messages from the clients are created through layered processing by the following message filters:

- **The SOAP Filter**, which constructs the SOAP envelopes and serialise the service messages following WSDL of the target services.
- **The Context Filter**, which serialises the operational context information to context elements and insert them into the SOAP headers. When necessary, it is also responsible to send detailed context information to the context service.
- **The Sequence Filter**, which serialises information for reliable messaging to *sequence* elements and acknowledgment requests, and put them in the SOAP headers.
- **The Security Filter**, which implements XML-Signature and XML-Encryption functions in WS-Security to sign and encrypt the SOAP messages using security tokens specified by the users.
- **The Dime Filter**, which handles the DIME [28] protocol that is used for delivering of large sets of data alongside the SOAP messages.

It is obvious that not all the filters are required for each operation on design optimisation services. The use of the filters and the usage sequence can be specified in the client configuration files.

4. Exemplars

In this section we demonstrate the use of our services deployed using technologies described in previous sections. Three categories of methods that are commonly used in engineering design optimisation are exposed in previous sections as services: design of experiments (DoE), response surface modelling (RSM), and genetic algorithm (GA). In engineering design optimisation, high-fidelity analysis codes such as FEM and CFD are often used to provide definitions of the measure of merits for the product. Due to the high computational cost of these high fidelity codes, a strategy of combining DoE/RSM is commonly applied to work together with evolutionary optimisation methods such as genetic algorithm.

In our exemplar, this strategy is applied to a numerical test function problem of a two-dimensional parameter space [29] to illustrate the use of our service-oriented design optimisation process orchestrated using the Matlab scripting language:

$$F(x) = \sum_{i=1}^{n}\Big(\sum_{j=1}^{n}(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j) - \sum_{j=1}^{n}(a_{ij}\sin x_j + b_{ij}\cos x_j)\Big)^2$$

Figure 9 shows the sample points in the design space generated by the DoE service, and response surface model built using the RSM service. Twenty points are initially used in the process of building the response surface. The GA service is subsequently invoked on the response surface model, and best points found on the response surface are evaluated using the true objective function, and response surface model are then updated using those points where the true objective function performs better than the response surface evaluation. These results are also shown in Figure 9.

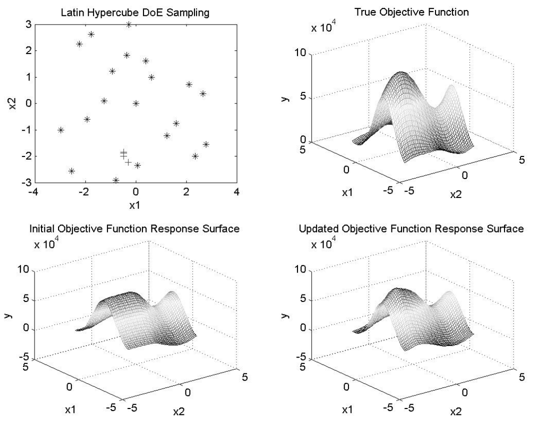


Figure 9: Application of Service-Oriented Design Optimisation Process for a Test Function

The second example involves a more complex engineering design problem. A parametric ProEngineer model of airfoil is used to generate different designs using the DoE services. After generating the design points, the geometry is evaluated in parallel on a Condor pool via our computational client service. Four of the geometries are shown in Figure 10.

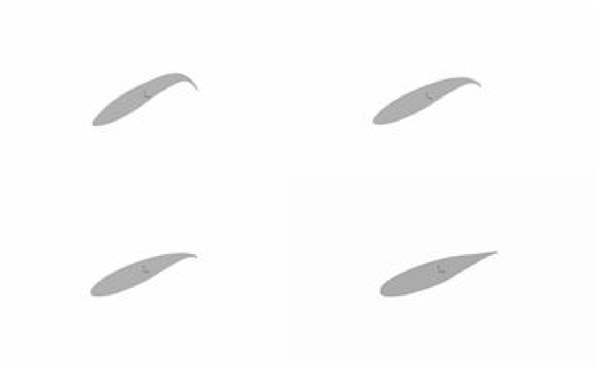


Figure 10: Four airfoil shapes produced by DoE service

5. Conclusions and Future Work

The demands for improved design optimisation processes have prompted our efforts on applying service-oriented Grid technologies to provide seamless system integration and access to additional computation resources. In this paper, we introduce our work in developing essential services that are used to construct the design optimisation process. The usage of advanced Web/Grid services technologies for supporting complicated design optimisation operations are explained in detail. Applications of our services in different stages of the design optimisation process are demonstrated with concrete examples.

Future work on the service development will focus on adapting the current implementation to new Web services architectures for Grid computing, such as the WS-Resource framework. We will also attempt to migrate the services to other Web services platforms to facilitate wider adoption.

Acknowledgement

This work is supported by the UK e-Science Pilot project (UK EPSRC GR/R67705/01).

References

[1] The Geodise Project. http://www.geodise.org

[2] Foster, I., Kesselman C., Nick, J., and Tuecke, S., “The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration,” 2002.

[3] Open Grid Services Infrastructure (OGSI), http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-29 2003 04 05.pdf, 2004

[4] Ian Foster,et. al. Modeling Stateful Resources with Web Services. http://www.globus.org/wsrf/ModelingState.pdf

[5] Keane, A. J., OPTIONS: Design Exploration System, http://www.soton.ac.uk/ ajk/options.ps, 2003

[6] Xue, G., Song, W., Cox, S. J., and Keane, A.J. Numerical Optimisation as Grid Services. Submitted to the Journal of Grid Computing.

[7] Song, W., Xue, G., Keane, A.J. and Cox, S. J. Implementation of a Genetic Algorithm as a Grid Service. Submitted to Europar 2004, Pisa, Italy.

[8] Xue, G., Fairman, M.J., Pound, G.E., and Cox, S.J. Implementation of a Grid Computation Toolkit for Design Optimisation with Matlab and Condor. Proceedings of Europar 2003, Klagenfurt, Austria.

[9] Wason, J.L, Molinari, M, Jiao, Z & Cox, S.J. Delivering Data Management for Engineers on the Grid. Proceedings of Europar 2003, Klagenfurt, Austria.

[10] Web Services Context (WS-Context) Specification. http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CTX.pdf

[11] The OGSI.NET project. http://www.cs.virginia.edu/~humphrey/GCG/ogsi.net.html

[12] The MS.NETGrid project. http://www.epcc.ed.ac.uk/~ogsanet/

[13] The Microsoft .NET Framework. http://msdn.microsoft.com/netframework/

[14] Web Services Architecture. http://www.w3.org/TR/ws-arch

[15] Service-Oriented Architecture (SOA) Definition. http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html

[16] Web Services Enhancement. http://msdn.microsoft.com/webservices/building/wse/default.aspx

[17] Web Services Reliable Messaging. http://www-106.ibm.com/developerworks/library/ws-rm/

[18] The WS-Security Specification. http://www.ibm.com/developerworks/library/ws-secure/

[19] Public Key Infrastructure. http://www.ietf.org/html.charters/pkix-charter.html

[20] Web Services Security Policy. http://www-106.ibm.com/developerworks/webservices/library/ws-secpol/

[21] Web Services Policy Framework. http://www-106.ibm.com/developerworks/library/ws-polfram/

[22] Wasson, G. and Humphrey, M. Attribute-Based Programming for Grid Services. 2003, GGF9 Workshop on Designing and Building Grid Services.

[23] Common Language Runtime (CLR) Overview. http://msdn.microsoft.com/library/en-us/cpguide/html/cpconCommonLanguageRuntimeOverview.asp

[24] PORT Mathematical Subroutine Library. http://www.netlib.org/port/

[25] Java Native Interface (JNI). http://java.sun.com/

[26] Microsoft COM Technologies. http://www.microsoft.com/com/

[27] Business Process Execution Language for Web Services. http://www-106.ibm.com/developerworks/library/ws-bpel/

[28] DIME. http://www.ietf.org/internet-drafts/draft-nielsen-dime-02.txt

[29] Schwefel, H.P. *Evolution and Optimum Seeking,* John Wiley & Sons (1995).