The GRID: Computational and Data Resource Sharing in Engineering Optimisation and Design Search

Cox SJ, Fairman, MJ, Xue, G, Wason, JL, [+]Keane, AJ
Department of Electronics and Computer Science
[+]School of Engineering Sciences
University of Southampton
{sc, mjf00r, gx00r, jlw98r}@ecs.soton.ac.uk, ajk@soton.ac.uk

Abstract

In this paper we present our GRID architecture for engineering optimisation and design search. We are developing a system that will allow seamless access to an intelligent knowledge repository, a state-of-the-art collection of optimisation and search tools, industrial strength analysis codes, and distributed computing and data resources. We focus on the underlying open standards technologies required to implement our system, and give exemplars of how they are being exploited at present. Of particular importance is the interchange of data throughout the system, for which we have adopted W3C standards (e.g. XML), and the ability to link together each of the components in the form of web services.

1. Introduction

Engineering optimisation and design search is the *process* whereby existing engineering modelling and analysis capabilities are exploited to yield improved designs. In the next 2-5 years intelligent search tools will become a vital component of all engineering design systems. Such facilities will steer the user through the process of setting up, executing and post-processing design search and optimisation activities in a variety of disciplines.

A major driving force in these developments is the need to allow distributed design teams from multiple enterprises to access design and analysis capabilities via the GRID. The GRID has been characterised by a 3 layer model which consists of (i) a computation / data grid, (ii) an information grid, and (iii) a knowledge grid (Figure 1.)

In this paper we focus on the technologies which we are using to implement our grid-based system. In section 2 we introduce the Grid architecture for our optimisation system. Section 3 discusses the open standards technologies which we are using. We discuss the computation, data, information and knowledge layers of the grid in sections 4-7 and draw our conclusions in section 8.

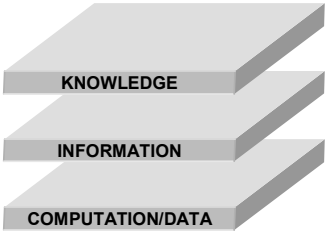

Figure 1: GRID Architecture

2. Optimisation and Design Search Grid

2.1. Architecture

Figure 2 shows our grid-based system. In a typical scenario of, for example, wing design, an engineer may couple together Computer Aided Design (CAD) tools, analysis codes for Computational Fluid Dynamics (CFD), or Finite Element Analysis (FEA), and tools for optimisation. The resulting sequence of computations may be performed on local or national machines, or on pay-per-use internet cluster resources e.g. [1].

One important new component in the higher levels of the GRID is the application of database technology, which has conventionally only been used to store information about the products designed using the system. Key questions in the process of, say a typical wing design optimisation, are: What previous designs have been explored and how can I develop them further? Which optimisation strategies are likely to prove effective? Which computational resources have the analysis codes I require, and when will my results be ready? Here there are new requirements for information and knowledge to be extracted from

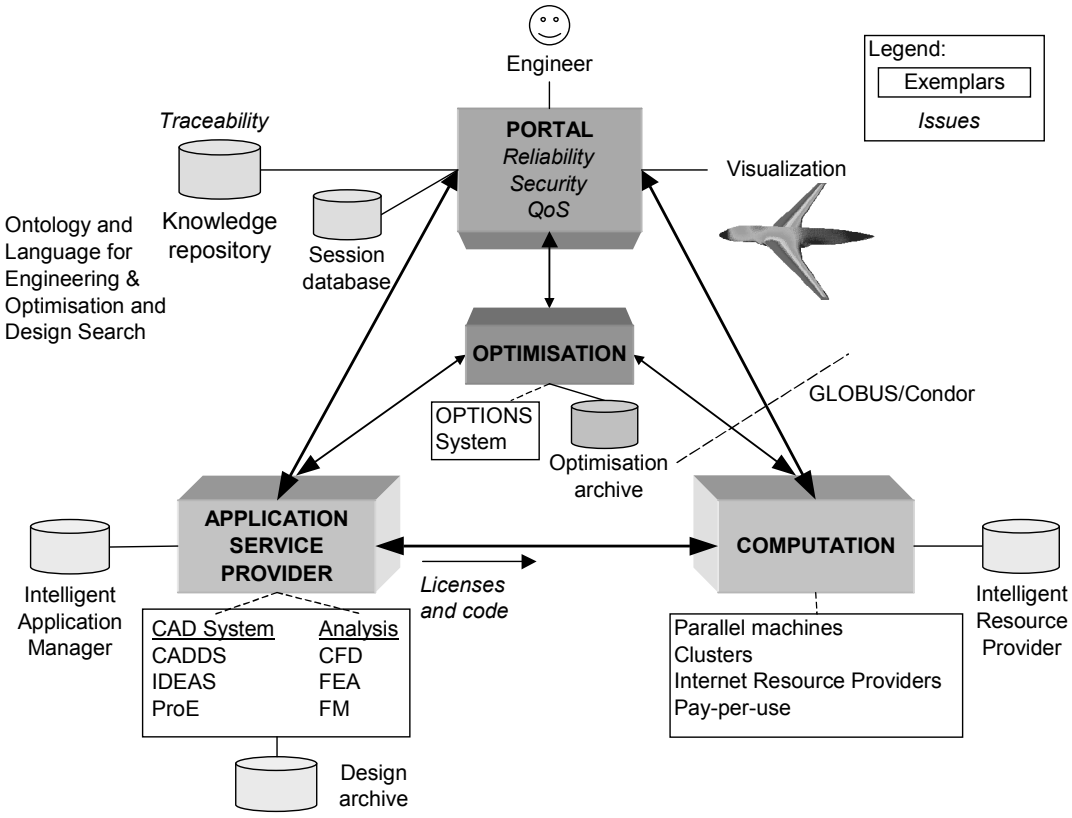

Figure 2: Engineering Optimisation and Design Search Architecture

automatically generated databases. Encapsulating and exploiting knowledge at all levels of the grid will enable new designs to be developed more rapidly, or at lower cost. Our system therefore includes a knowledge repository which may be queried.

The architecture of the system consists of four main components which we now discuss in turn.

2.2. Portal

This is the web-based point of access to the system, which allows engineers to locate and combine the services they require, giving advice as necessary. It provides grid-based seamless access to an intelligent knowledge repository, a state-of-the-art collection of optimisation and search tools, industrial strength analysis codes, and distributed computing and data resources.

It provides the information capabilities of a system like Tambis [2] which "*aims to aid researchers in biological science by providing a single access point for biological information sources round the world. The access point will be a single interface (via the World Wide Web) which acts as a single information source. It will find appropriate sources of information for user queries and phrase the user questions for each source, returning the results in a consistent manner which will include details of the information source*."

The service portal has the following roles:

? The engineer interacts with the system through the service portal.
? To advise the engineer which optimisations work well with which applications on which resources.
? It houses an intelligent knowledge repository to determine which web service databases contain the information required to answer questions it is posed. The knowledge base is built up over time in a manner similar to that used to cache queries on web search engines.
? To provide *traceability* and *security* for results produced on the system
? To negotiate *quality of service* (QoS) and *reliability* guarantees with other service providers.

2.3. Application Service Provider

The engineer requires access to both design and analysis tools, supported by a database which will provide information about previous designs. Whilst some tools may run adequately as a user application across a server, commercial compute-intensive analysis codes should be also available on a pay-per-use basis to run on external computational facilities provided by the resource provider. An early exemplar of this was developed at Southampton [3].

2.4. Optimisation service provider

The optimisation technology is provided by the OPTIONS [4] system, which has been continually developed over the last 15 years. OPTIONS is used for design optimisation and includes a variety (over 40 at present) of different optimisation algorithms.

At the start of the optimisation process, the engineer needs to provide initialisation information– in particular the analysis codes to be coupled together, the permitted methods by which a design may be modified, and an objective function by which each design may be evaluated. The optimisation service will then coordinate the maximization (or minimization) of the objective function to improve the design.

In our current implementation, the optimisation service has a knowledge repository of recommended settings to use for each optimisation control variable. Depending on the level of automation, the optimisation service may automatically enter or suggest to the engineer suitable values to use. The optimisation service then retrieves the required program from the application provider and executes it on computation resources provided by the resource provider.

The optimisation process is divided into several 'stages', which require single or multiple evaluations of the objective function. Each stage is a transactional unit and after successful completion the optimisation's related state and current data is automatically archived. Our framework fits into the model of web orientated application architectures such as Sun Microsystem's Java Enterprise Server where each optimisation code would be an Enterprise Java Bean. Since each step is a transactional unit we can provide consistency and durability and allow for the possibility of multiple independent objective function evaluations to be made in a naturally parallel way.

Furthermore since all data is stored along with session identity information it is flexible enough to meet the requirements of different requesters and can deal with multiple requests at the same time.

2.5. Resource Provider

The resource provider's main role is to execute code to return the value of the objective function which is being optimised. It has a database of computational resources which it provides and may in addition need to negotiate any licences required from the application service provider when commercial code is used.

2.6. Business Process

In the process of service integration and collaboration, the service providers and requesters must come to an agreement about how the interactions will be conducted. This is done through negotiation between the two sides. To carry out the negotiations, both sides should have a common protocol to understand messages from the other side and access to a knowledge base to inform the decision process.

A computer resource in a large-scale computational environment is expected to have such negotiations and interactions with different kinds of service requesters. To make a true sharing and integration of the resources, it is necessary to have a standard business process - the knowledge for negotiating and interacting - for all web services and service consumers.

The business process model for web service integration in an engineering design optimisation environment can be abstracted from practical experience in various applications of web services. A standard workflow for interactions is to be built as well as a framework for message deliveries. In addition, the model should be able to adapt itself to meet new demands from the system. This requires an extensible workflow and message framework.

The BizTalk system brought forward by Microsoft [5] provides a good reference to the design of a business process model for the engineering design optimisation system. BizTalk brings a whole solution to carry out e-business among different web services providing tools for defining business workflows and message framework. It is also scalable and flexible to meet specific requirements and future extension.

3. Open Standards Technology

Engineering design optimisation requires various computational and data resources to be integrated. To achieve this, technical obstacles brought about by the differences between system environments, software and programming languages must be overcome. One solution to this problem is to wrap various computer systems in a common interface, which 'talks' with the others in a standard format. The recent progress in several XML-based open standard technologies [6] has brought major advances in this approach. These contrast with traditional Java/ Jini based approaches by adopting very simple lightweight protocols. Figure 3 shows how the technologies discussed in the next sections are related to each other.

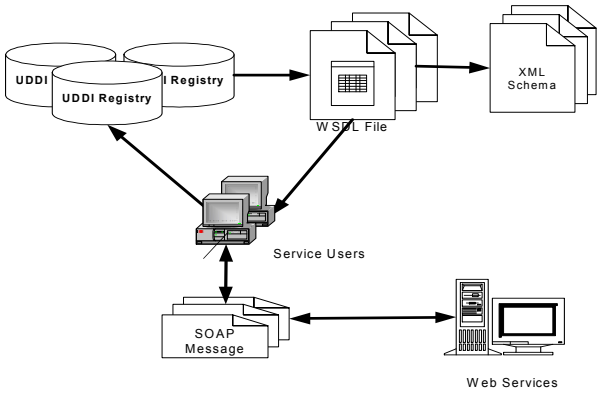

Figure 3: Use of XML technologies to provide web services

3.1. XML and XML Schema

XML provides a method to contain data of structured format in plain text. Data in the form of XML are transferable between almost any computer system and software application that have the ability to read plain text. XML, as plain text, also works well with existing Internet Protocols like HTTP or SMTP. Thus XML is an ideal medium for communication between different computer systems. Furthermore security can be provided by tunnelling the XML through a secure layer such as SSL [7].

XML Schema is a W3C alternative specification to the Document Type Definition (DTD) for describing the structure of an XML instance document. Compared to the DTD, XML Schema is both more powerful, incorporating a set of rich data types, and more flexible, supporting sophisticated user-defined data structures. So with XML Schema, it is possible to use XML to reflect the complexity of the existing computer resources. Also by using XML Schema, validation of data is transferred from the application role to the parser, thus making the separation between data logic and application logic possible.

3.2. SOAP and XML Protocol

The Simple Object Access Protocol (SOAP) [8], provides a simple and extensible framework to define how an XML message is structured. It is a lightweight protocol for the exchange of information in a decentralized, distributed environment and is easily carried via various Internet protocols, such as HTTP and SMTP. SOAP allows security systems like a firewall to identify an XML message without needing to understand its contents, thus it is feasible to prevent the blocking of unknown HTTP requests. SOAP also provides rich semantics for indicating encoding style, array structure, and data types. SOAP is currently under inspection by the W3C consortium and is a prototype of the future XML Protocol.

3.3. Web Service and WSDL

With XML Schema, SOAP and their supporting software, it is feasible to wrap various computer resources into similar XML-interfaced web components, which are called web services. These communicate with each other using XML messages. To access a web service properly, description about the service interface is required.

Web Service Description Language (WSDL) is one of the emerging technologies for this purpose. WSDL, written in XML, describes a web service as a set of endpoints. Each endpoint consists of several operations and is bound to a specific network protocol (e.g. HTTP or SMTP). The input and output messages are defined for an operation and XML Schema is used in the WSDL file to provide the data types and structures to define the messages. Together with the XML Schema, it provides a complete definition for the interface of a web service and allows programmatical access to the service, in the manner of an API. Tasks like data requests or execution of a piece of code are then performed by sending and receiving XML messages using the SOAP mechanism. For the publishing, discovery and integration of a web service, additional technologies are needed.

3.4. UDDI

The Universal Description, Discovery and Integration (UDDI) specification defines a way to publish and discover information about web services [9]. It is a collaboration between Ariba, IBM and Microsoft who each provide UDDI services. The UDDI project includes a UDDI business registry and a set of operations on it. The UDDI registry, an XML file, identifies a web service and provides information about the service. Other programs use the registry to get the information about the web service and check compatibility with it. Categorisation of web services in the registry enables location and discovery. UDDI together with WSDL, provides the ability to locate and programmatically interface to the web service. This allows the web service to be used in a program in a similar way to a software component, and hence simplifies the service collaboration.

We now discuss the levels of the grid in more detail.

4. GRID: Computation

This section considers the concept of resource management and computational resources. Condor [10] is used as an example of an existing resource sharing system. It is chosen because its concepts and architecture are well suited to the GRID model and are applicable to other and new resource sharing systems. We demonstrate how it can be wrapped and offered as a generic web service.

4.1. Condor Introduction

Condor is a software system that creates a High-Throughput Computing (HTC) environment by harnessing the power of UNIX and NT clusters and workstations. It can manage dedicated clusters however its main appeal is that it is able to use pre-existing resources in a distributed ownership context where computers are sitting on people's desks.

When jobs are submitted to Condor it finds an available machine in its organisation's pool to run the job. Machines become available once they have been idle for a specified period for example when the owner goes to lunch. Jobs are migrated over the network to the machine. If the machine becomes unavailable and the job has not finished (such as if a user returns to their desk after lunch), Condor checkpoints it and either migrates the job to another machine or queues it to disk until a machine becomes free.

4.2. Resource Management

Condor provides a distributed resource management system, which matches the resource consumer's needs with the resource owners. Instead of attaching properties to a job queue directly, Condor implements a publishing system where each machine advertises their resource properties called ClassAds. Each submitted job specifies a resource request ad, which defines the required and preferred run properties of the job. Condor performs brokering by matching and ranking the ClassAds with a job's resource requests.

4.3. Condor and the GRID

Whilst we treat Condor as fundamentally a resource provider at the lowest level of the GRID, its overall architecture fits into the component layers of the GRID model. If we view the model from the bottom up, at the computation resource level are the machines in Condor's pool with data and control between machines and the Condor control daemons being performed through Remote Procedure Calls. Each machine's ClassAds publishing provides resource description and discovery at the information layer. Condor provides the resource integration of the information layer with its resource management and migration. At the knowledge layer is the resource manager, which performs sophisticated resource brokering.

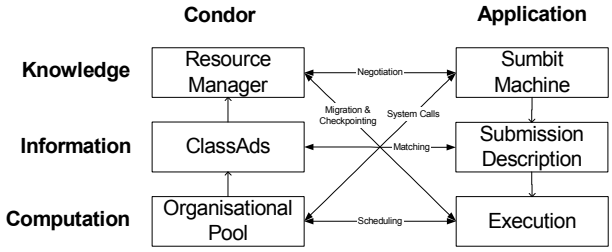

Figure 4: Condor GRID Model

There are however some fundamental differences between Condor and the GRID. As a project it began development before the concept of the GRID came into existence. As such parts of the system, for example the communication system use older technologies (RPC) and Condor uses its own proprietary systems for resource description, discovery and integration. Furthermore, unlike the components in our GRID architecture, Condor does not expose a programmatic interface as a web service: it is more UNIX-like in that interaction between users and Condor are through the command line.

4.4. Condor as a Web Service

It is feasible to integrate Condor into the GRID because of its clear separation of its computational and information layers. We now show how it is possible to expose Condor's resource management ability programmatically. Using Microsoft's .NET framework [11] and Visual Studio.NET, Condor has been successfully wrapped as a C# web service.

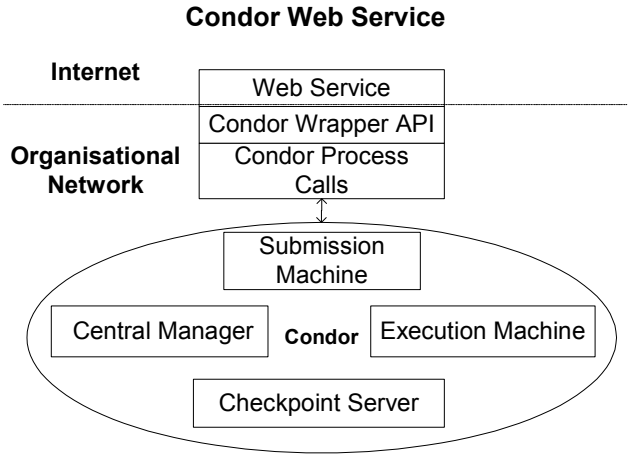

Figure 5: Condor Web Service

4.5. Legacy System Integration

The two most important Condor programs are `condor_submit` and `condor_status`. The first provides the user with the ability to monitor and query the machine pool. Submision of jobs is performed by the `condor_submit` command, which requires a

submit-description file. Other commands provide control, monitoring and querying of submitted jobs.

It is possible to launch any command line executable as a process from another program. Input to the command line executable is through the standard input, read files, pipes, and process start properties. Output is retrieved from the standard output, standard error, written files, and pipes. Construction of an API is then achieved using standard IO and process function calls.

Code wrapped around Condor by modelling each of its programs as a process, facilitates the creation of an API. It then becomes possible to make use of these APIs in the web service code to expose a programmatic interface to Condor on the Internet.

ClassAds and Submission Description files (Job Requirement descriptors) are flat files, which are translatable to and from XML documents and have structures that are directly describable as XML Schema documents, as shown in Figure 6.

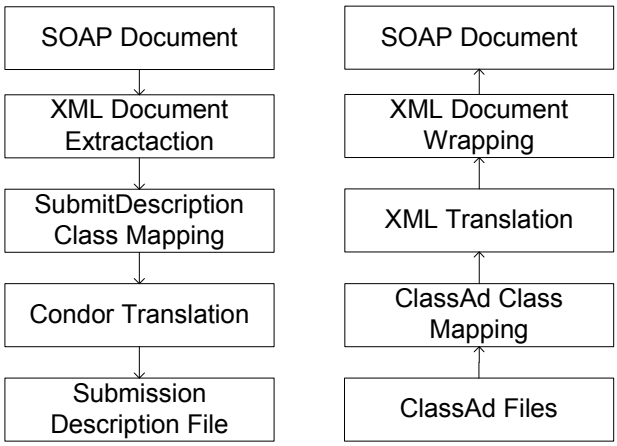

Figure 6: XML Wrapping Process

WSDL provides a standard way for the Condor web service to describe its abilities. Contained in the WSDL document is Condor's functionality, method of interaction (SOAP), job sessions, and the structure of the data types it exchanges.

4.6. Submitting Jobs

Submitting jobs to Condor requires both a submission description file and an executable. Condor allows heterogeneous submission of executables on differing platforms. No special coding is required; the only restriction is that to support check pointing, the code must be statically re-linked with Condor's libraries. Where this is not possible e.g. in the case where the source code is not available, then check pointing is not available and the executable must run until it completes, fails or is stopped by Condor to allow the resource owner to use their machine.

Condor uses the concept of universes to cater for the differing execution environments. For non-parallel code, there are the standard and vanilla universes, supporting re-linkable and non re-linkable code respectively. The MPI, PVM and Globus universes give support for parallel code.

From a commercial usage point of view, the vanilla universe is ideal in that the program supplier does not have to reveal the source code. When using Condor as a web service, the most basic level of executable environment is the vanilla universe because it requires the resource consumer to perform the compilation and linking. The main disadvantage of the vanilla universe, from the web service provider point of view, is that to provide an adequate level of service, a dedicated pool of executing machines is necessary because otherwise jobs would be regularly killed as owners used their machine for other purposes.

Where an organisation has a large number of non-dedicated machines it wishes to cycle steal from, then the web service must be capable of supporting re-linking of source code. It is possible to run an executable in the standard universe which has be linked by the resource consumer, however this is not ideal as it requires Condor's libraries and compiler. We have been concentrating on Condor but it is not the only resource management system. Is is desirable that a resource consumer need no knowledge of internal workings of a resource management service otherwise each consumer would need to support every resource managers method of linking. Instead the web service should describe in its WSDL the supported programming languages in its standard universe and perform the compilation itself.

4.7. Separation of Management from Computational Resource

It is desirable to be able to separate Condor's resource manager system from the resource pool. Condor's architecture requires that each machine within the system (including submission machines) run Condor control daemons, which communicate with a central manager and the submission machine through RPC. This architecture is not suited to large numbers of distributed computational resources on an Internet scale. While the problem of remote submission of jobs from an non-Condor machine is possible through legacy system integration (section 4.5), it is difficult to add machines to Condor's organisational pool which are not on a local network because of the usage of RPC and the possibility of firewalls. Also machines in the pool are available only to Condor and a single Condor central manager (i.e. there is no sharing among Condor systems.)

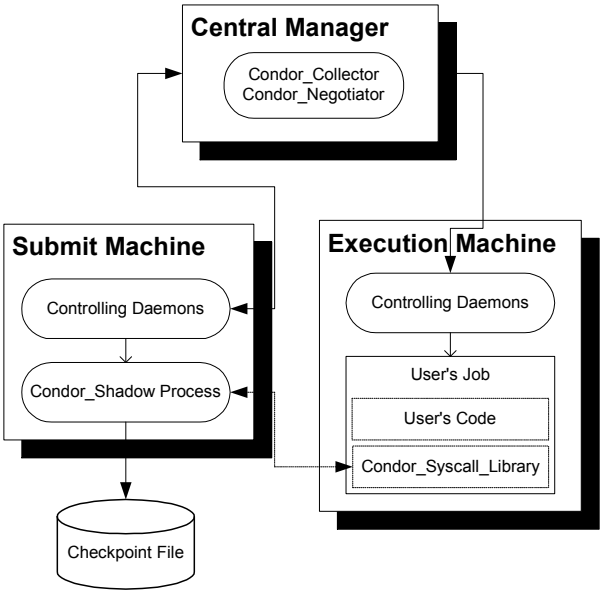

Figure 7: Condor Pool Architecture

Separation of resource manager from computational resources would allow sharing of resources amongst brokers, while at the same time providing an architecture that is capable of achieving a much more scalable system of distributed computational resources. With this it becomes possible to introduce a business process at the level of computational resources where each resource could charge for computation time used. The resource manager and computational resources become separate services on the Internet each with a defined role and business process.

4.8. Computation Issues

At the knowledge level sits the resource manager, which provides a brokering service to match resource requests with resource offers. However, now that each computational resource is separate from any particular resource manager, a user or other GRID service may wish to access the computational resource directly or in combination with the resource manager. This creates an environment where resource managers and computational resources compete to provide superior services and lower costs.

A question not answered here is support for check pointing and migration. If computational resources are to be shared among resource managers, then it is necessary to define a standard mechanism for this. Who does the linking, the resource manager or the computation resource, or neither? What about code security and how to prevent malicious or bad code from effecting a computational resource or interfering with another's code? A solution to these issues may be to use a platform independent programming solution such as Java [12] or the Microsoft Common Language Runtime System [13], which both run code in a sand box environment with a definable access policy.

5. GRID: Data

5.1. Metadata

Data is an essential part of the optimisation and design search process. Integration of the Grid relies heavily on the data exchanged between various computer systems. Definitions of types and structures of the data are required to ensure a correct operation. The web services use XML for data exchange. So it is necessary to provide a generic, cross-platform method to define the XML data types and structures.

The solution is still XML, or more specifically, XML Schema. XML Schema uses XML itself to define the types and structures of XML data. It provides rich support for basic data types like integer and string as well as common data structures available in computer programming languages. It is also possible to construct user defined data formats, such as postcodes.

The flexibility of XML enables the XML Schema to support the sophisticated data structures necessary to define complex user types for web services. As an example, a reply to the Condor_Status operation is to bring a list of "ClassAd"s consists of two attributes: "Arch" and "OpSys", corresponding to the system architecture and operating system, which are assigned specific values. A fragment of the XML Schema describing the message is shown in Figure 8.

Another advantage of using XML Schema as the data type and structure information provider is that XML Schema works with the XML parser, rather than the application. This provides a separation between the application logic and the data logic, which simplifies (and reduces) any maintenance required as a result of changes on either the data or application side.

XML Schema is designed to be reusable. A new piece of schema can be built based on several existing ones. This brings more efficiency and basic elements for a knowledge system. There have already been several services for publishing XML Schemas on the Internet, (such as BizTalk.org).

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
            elementFormDefault="qualified">
    <!-- The other pars... -->
    <xsd:element name="Condor_State">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="ClassAd" type="ClassAdType" minOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="ClassAdType">
        <xsd:sequence>
            <xsd:element name="Arch" type="Architecture" minOccurs="1"
                         maxOccurs="1"/>
            <xsd:element name="OpSys" type="OperatingSystem" minOccurs="1"
                         maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="Architecture">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Intel"/>
            <xsd:enumeration value="Sun"/>
            <!-- and so on ... -->
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="OperatingSystem">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Windows"/>
            <xsd:enumeration value="Linux"/>
            <!-- and so on ... -->
        </xsd:restriction>
    </xsd:simpleType>

    <!-- The other pars... -->
</xsd:schema>
```

Figure 8: Example XMLSchema Listing

5.2. Role of Databases

Knowledge, information, data and computation exist at various points in the architecture; for example, intelligent decisions must be made to carry out resource management. It is desirable to database information at each web service for the following reasons:

1. Each web service (resources, application service provision and optimisation) can use the archive itself to make intelligent decisions about its own function.
2. The entry portal may query these databases to build a knowledge archive for the system. Queries may be offered as part of the WSDL specification for the service.
3. Information about users, times, machines, optimisation strategies and results can be transparently logged by the web service as XML conforming to an XML Schema and automatically archived using JDBC [14, 15]. If a database does not yet exist for this data it can be generated automatically from the XML Schema. When the structure of the XML Schema changes, for example if a new optimisation method is added; the database must also evolve– we are currently working on this.

5.3. Automated Databasing Methods

Each web service is described by a WSDL document (see section 3.3) which includes an XML Schema to describe the data types in the SOAP messages. We have developed methods to generate a new database automatically from the XML Schema contained in a web service's WSDL [14]. Subsequently, any XML request and response data can be transparently logged and archived in this database, using the XML Schema as a guide.

The rich data type and structural support of XML Schema makes it a good candidate for automatic conversion to a database schema, and its original language requirements specify a type system adequate for import/export from database systems [16]. The fact that it is also defined in XML means it can be parsed using existing, generic tools. Other features such as the ability to define default values, scoped unique values, keys and relationships can also be employed for use with databases.

Relational databases can be manipulated with SQL using JDBC, which a Java API for accessing databases which is independent of vendor and platform. We have implemented a set of tools for database generation and data storage using an XML Schema with a combination of JDBC, XML parsing, and some conversion rules. Due to the component like nature of our architecture the data could also be stored in an object-oriented or XML database, so long as the equivalent schema generation and archiving tools were implemented.

As a simple relational database example, the XML Schema fragment in Figure 8 can be translated into a table called `ClassAd` with two columns of type `varchar` called `Arch` and `OpSys`. Each of these columns has a check constraint to ensure the validity of its data values, for example column `Arch` can only contain one of the values 'Intel', 'Sun', etc. However, it must be noted that these check constraints are unlikely to fail, as the parser will already have validated data against the XML Schema before it is automatically archived. The table will also include a column to uniquely identify a record, such as a session ID, from further up in the XML Schema hierarchy.

6. GRID: Information Layer

We have shown that systems like Condor for managing computer resources can be exposed and offered as a web service accessible to applications on the internet. However, to realize a true integration of the GRID, additional information about the web services is required. This divides into three elements; a definition of data types and structures, a description of the service interfaces, and the categorization information of the web services. Together these three elements comprise the information layer of the GRID model. Figure 9 shows how the open standards described in section 3 provide the functionality we require.

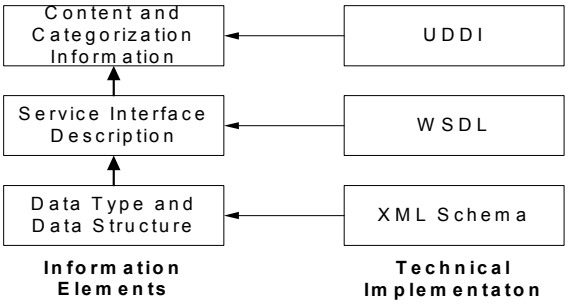

Figure 9: Information Elements

7. GRID: Knowledge

The information layer adds meaning to data and computation and the ability to access it programmatically. The knowledge layer provides an intelligent problem solving environment to guide the engineer through the process of combining the different components required to perform optimisation and design search efficiently. The intelligent knowledge repository is built by querying the archives of each web service from which, for example, new heuristics about optimisation can be deduced and used to improve future design processes. This may be achieved using formal knowledge elicitation methods and held in rule bases. Design activity also creates domain specific knowledge that designers may wish to archive and reuse in various ways to enhance their design capabilities in the future. Various knowledge management and re-use tools can be used to underpin this process.

8. Conclusions

We have demonstrated that open standards can be used at all layers of the GRID to provide transparent access to applications, optimisation codes, distributed computational resources and knowledge repositories. We view each of the components in the GRID model as web services which may be integrated together to provide, in our case, the framework for an engineering optimisation and design search system. In particular we have demonstrated how the Condor system for exploiting idle compute cycles can be offered as a web service and how automated databasing techniques fit naturally into the web service architecture.

9. References

[1] http://www.entropia.com/

[2] P.G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, R. Stevens, TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. An Overview in Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology, ISMB98, Montreal, 1998
http://img.cs.man.ac.uk/tambis/

[3] M J Addis, P J Allen, M Surridge. Negotiating for Software Services (2000) Proc. IEEE 11th International Workshop on Database and Expert System Applications p.1039-43 (DISTAL project)

[4] Keane, A.J. (1999) The Options Design Exploration System – Reference Manual and User Guide, Computational Engineering and Design Centre, University of Southampton.
http://www.soton.ac.uk/~ajk/options.ps
http://www.soton.ac.uk/~ajk/options/welcome.html

[5]
http://www.microsoft.com/biztalk/productdoc/framework20.html

[6] W3C - World Wide Web Consortium (1998). Extensible Markup Language (XML) 1.0. Editors Bray, T., Paoli, J & Sperberg-McQueen, C.M
http://www.w3.org/TR/REC-xml

[7] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL protocol version 3.0. Netscape, Transport Layer Security Working Group, November 1996
http://home.netscape.com/eng/ssl3/draft302.txt
http://www.netscape.com/eng/ssl3/

[8] Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000
http://www.w3.org/TR/SOAP/

[9] Ariba, IBM, Microsoft. UDDI Executive White Paper, 6th Sept 2000. From:
http://www.uddi.org

[10] J Basney and M Livny, Deploying a High Throughput Computing Cluster, in *High Performance Cluster Computing*, Rajkumar Buyya, Editor, Vol. 1, Chapter 5, Prentice Hall PTR, May 1999.
http://www.cs.wisc.edu/condor/

[11]
http://msdn.microsoft.com/net/framework/default.asp

[12] Java TM 2 Platform, Standard Edition White Papers
http://java.sun.com/j2se/1.3/white.html

[13]
http://msdn.microsoft.com/library/dotnet/cpguide/cpconwhatiscommonlanguageruntime.htm

[14] JL Wason, SJ Cox, and AJ Keane, "Flexible Knowledge Repositories for Problem Solving Environments," pp. 199-205 in Proc. Int. Workshop on Advanced Data Storage / Management Techniques for High Performance Computing, ed. R. Allan and K. Kleese,CLRC, Daresbury, ISSN 1362-0223 (2000).

[15] Sun (2000) JDBC TM Data Access API, Sun Microsystems, Inc.
http://java.sun.com/products/jdbc/index.html

[16] Malhotra, A. & Malonay, M. (editors) (1999) XML Schema Requirements, *W3C NOTE*, NOTE-xml-schema-req-19990215, February 1999
http://www.w3.org/TR/NOTE-xml-schema-req