A Service-Oriented Approach for Aerodynamic Shape Optimization across Institutional Boundaries

Wenbin Song[1], Yew Soon Ong[2], Hee Khiang Ng[2], Andy Keane[1], Simon Cox[1], and Bu Sung Lee[2]
[1] Southampton e-Science Centre, University of Southampton, Southampton, SO17 1BJ, UK,
{w.song, ajk, sjc}@soton.ac.uk
[2]School of Computer Engineering, Nanyang Technological University, Singapore 639798
{asysong, mhkng, ebslee}@ntu.edu.sg

Abstract

This paper presents the experiences gained from ongoing research collaboration between the School of Computer Engineering at Nanyang Technological University and the Southampton e-Science centre at the University of Southampton using a service-oriented approach for complex engineering design optimization. The service-oriented approach enables programmatic collaboration to be realized while maintaining the autonomy of individual codes at the different institutes and organizations. In the current work, a Genetic algorithm optimization logic implemented as a Grid service at Southampton is used to drive the design search process, while the aerodynamic analysis code located in Singapore is used to evaluate the objective function of the design points. Experience gathered from the current study on airfoil shape optimization is valuable for establishing effective, efficient and customised programmatic links between institutions to solve complicated engineering design problems.

1. Introduction

Collaboration lies at the heart of Grid computing. [1] Grid computing provides the infrastructure and tools to encourage collaboration between departments, institutions and research centres to deliver better research outcomes. The concepts of service-oriented computing are seeing wide acceptance and becoming a standard approach for tackling distributed, heterogeneous computing problems. Web services/Grid services are evolving fast, and the proposed WSRF (WS-Resources Framework) [2] is seen as a joint effort between industry and academia and a sign of the convergence of Web services and Grid Services. The principle motivation for adopting web services is interoperability, where a service consumer can access Web services using standard protocols such as SOAP (Simple Object Access Protocol) [3] and HTTP, etc., irrespective of the operating systems and programming languages that either parties use. Therefore, it increases the accessibility of existing capabilities by presenting existing codes as Web services and enables collaborations across geographic and institutional boundaries.

Numerical optimization methods are often used in the engineering design process to improve product performance and quality over a shorter design-cycle time. In many complex engineering design problems, high-fidelity analysis models are employed where each function evaluation may require a Computational Structural Mechanics (CSM), Computational Fluid Dynamics (CFD) or Computational Electromagnetics (CEM) simulation costing minutes to hours of supercomputer time. A motivating example for us is aerodynamic shape design of airfoils. The design optimization approach typically involves many repetitive function calls to the high-fidelity analysis codes to obtain the merits of the different combinations of design variables. This process is therefore both computational expensive and data intensive.

To increase the robustness and probability of obtaining a globally improved design, stochastic techniques such as evolutionary algorithms (EAs) [4] are often adopted as the principle algorithm in the design search process. Evolutionary algorithms, unlike conventional gradient-based numerical optimization methods produce new design points that do not use information about the local slope of the search function and are thus not prone to stalling at local optima. They have shown considerable success in locating the global optimum solution of many optimization problems characterized by non-convex and disjoint solution spaces. However, this increase in capacity to locate the global optimum designs is often achieved at the expense of greater computational cost and hence a longer design cycle time. Since EAs typically require thousands of function evaluations to locate a near optimal solution, the use of EAs such as Genetic algorithms (GAs) in general increases the cost of finding globally optimum designs.

This poses a serious impediment to the practical application of combining high-fidelity analysis codes and evolutionary design optimization in complex design problems in science and engineering. It is thus important to retain the appeal of evolutionary design optimization algorithms and to effectively handle computationally expensive design problems in order to produce high quality designs under tight design time schedules.

Fortunately, a well-known strength of EAs is their ability to partition the population of individuals among

multiple compute nodes. Since the design optimization cycle time is directly proportional to the number of calls to the analysis solvers, an intuitive way to reduce the total search time of evolutionary optimization algorithms is to parallelize the analysis of the design points. All design points within a single EA population may be evaluated simultaneously across multiple compute nodes.

The benefits of combining conventional parallel processing capability and the emerging Grid computing technology in the context of evolutionary design optimization are numerous. In particular, the Grid provides the infrastructure to facilitate distributed computing and parallelisms by tapping on vast compute power and a secure means of solving large-scale optimization problems. In this paper, we consider using Web services to provide access to high fidelity computational fluid dynamics analysis codes and compute resources located at Singapore and Grid services at Southampton to provide the optimization logic of a genetic algorithm to drive the design search process.

The rest of this paper is organized as follows: In section 2, we present a brief overview of service-oriented computing. Sections 3 and 4 describe the implementation aspects of wrapping airfoil analysis code as services and realizing genetic algorithms as Grid services, respectively. Section 5 illustrates the consumption of the services within the Matlab environment. Section 6 presents the experimental studies and discussions while section 7 concludes this paper.

2. Towards Service-Oriented Computing

Service-oriented computing (SOC), an emerging paradigm for distributed computing, represents a major shift in the way that software systems are designed, developed, and consumed. Services are platform and language independent software elements that can be described, published, orchestrated using XML data.

Web services [5] provide the programmatic interfaces between diverse applications via the Web. The WSDL [6] is used to describe the network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The introduction of stateful Web services leads to the development of Open Grid Services Architecture (OSGA) and associated implementation OGSI (Globus 3.0). [7]

In addition to building Web services interfaces to existing applications, there is also a need for a standard approach to connect services together to create constructive and effective customized processes. Web services orchestration [8] is about providing an open, standards-based approach for connecting web services together to create advanced processes.

3. Airfoil Aerodynamic Analysis as Web Service

The collaboration setup between the University of Southampton (Soton) and Nanyang Technological University (NTU) employs a multi-tiered service architecture as illustrated in Figure 1. The architecture consists of a Client tier, at Soton, a Web service tier at NTU, and the Grid resource tier utilizing the NTU campus grid [9] resources. The client tier at Southampton uses Java clients running within Matlab/Jython to send requests to the aerodynamic analysis service tier located at NTU. This tier thus provides the access point to the airfoil code which will then access NTU campus grid resources to carry out the computation.

In regard to security, we have used Globus GSI [10] enabled Web services to provide the essential security for the airfoil analysis Web services we deployed. The client generates the necessary proxy certificate from a grid proxy initialization process through the user's GSI certificate and private key. This proxy credential is then sent to the Web service using the HTTPG [7] protocol. The Web service will extract the credential context from the service request to authenticate the user and using this delegated credential to authorize access to the grid resources. Figure 2 illustrates this GSI web service architecture.

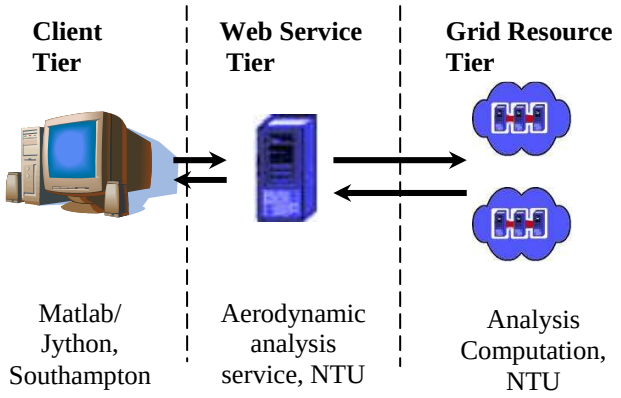


Figure 1: Collaboration Architecture

At the grid resource tier, the GRAM's [11] gatekeeper of the resource cluster upon receiving the job submission service request through the Globus web service spawns the aerodynamic analysis jobs across multiple computing nodes in parallel. In the process, the design input files are uploaded and transferred to the grid resource clusters. This is accomplished via the upload file web services provided. First, the input files are uploaded to the service provider web server using the file upload service. Subsequently, the GridFTP [12] web service, which is designed to handle peer to peer transfer of data files, relocates the files to the chosen compute nodes to proceed with the analysis computation. Upon completion, the results are marshalled back to the Globus web service via the Global Access to Secondary Storage (GASS) [13]. The

responsibility of the agent is to perform local scheduling and resource discovery across the computing nodes in the cluster. A brief overview of the airfoil analysis execution process is given in Figure 3.

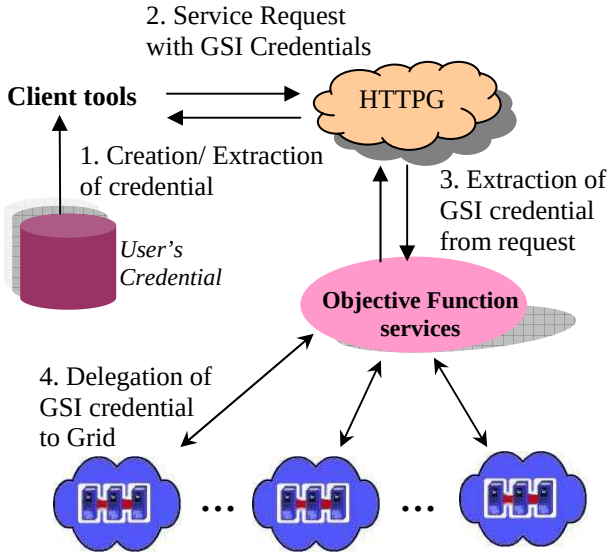


Figure 2: GSI Web Service Architecture

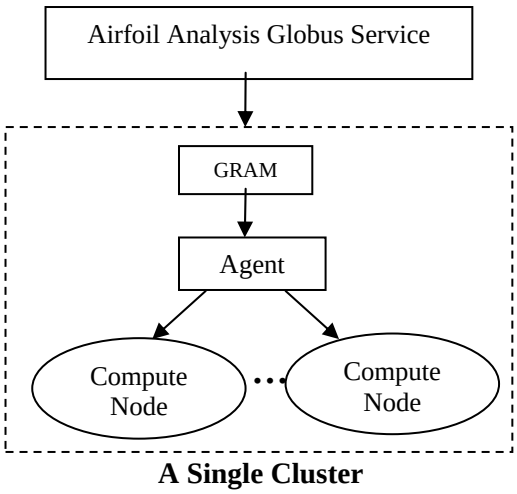


Figure 3: Airfoil analysis execution process

4. A Genetic Algorithm as a Grid Service

The growing popularity of evolutionary computation algorithms such as genetic algorithms [4] in design optimization has resulted in many implementations of the basic algorithm. These include some well-established and validated implementations, for example, one such implementation in the design exploration system OPTIONS [14]. The Genetic algorithm implemented in OPTIONS is typical of those discussed in the literature except that it incorporates a version of an adaptive clustering algorithm based on K-mean distance, and an elitist survival strategy. For constrained optimization problems, two types of penalty functions can be used when calculating the fitness of the design.

Access to the algorithms is not only limited to the availability of the code, but also the various types of interfaces used by the codes, including platforms, languages, etc. This presents further challenges when used in a wider scope for solving complex engineering problems, as it is often the case that various software components need to be integrated to create customized workflows or tools to suit a particular problem. Although this can be achieved by using a dedicated GUI-based environment such as iSIGHT [15] or ModelCenter [16], or using some free-form scripting languages such as Jython [17] or Matlab [18], it is usually beneficial to expose the optimization logic in a standard interface independent of the implementation languages and platforms, thus relieving the difficulties of integrating various components.

In the current work, our in-house design exploration system OPTIONS developed using Fortran 77 over the last 20 years is used to provide the optimization logic behind the optimization Grid service. OPTIONS maintains its own internal data structure which allows users to establish an optimization problem and manipulate the definitions of variables through a Fortran/C API. However, the Fortran/C interface it uses to communicate to users' simulation codes requires users to wrap their simulation codes as Fortran/C subroutines and link them with the system libraries. Alternatively, standalone packages may be executed as system calls. With the increasing complexity and the large number of simulation packages that are often involved in a typical multidisciplinary design optimization process, more advanced and flexible integration frameworks are required to couple users' codes with the optimiser. The efficiency and ease-of-use offered by more recently developed programming languages such as Java or C# and the open standards base Grid services should be exploited to enhance present engineering design processes.

4.1 Wrapping Native code into a .NET Managed code

The implementation of the genetic algorithm as a grid services is here presented as two phases. The first phase describes the steps that are required to prepare the code written in Fortran for use in other more network service-aware programming environments. The approach adopted can be used for generic cases where legacy codes (written in Fortran or C) need to be accessed from these environments.

Here, we target on shared libraries for Windows and Linux operating systems. Two steps are involved in this phase because the Win32 shared library generated by a conventional Fortran compiler such as Compaq Visual Fortran (CVF 6.1.0) [19] cannot be directly deployed as .NET applications. Here, a Salford FTN95 Fortran compiler [20] is used in place to create a library which itself calls routines in the Win32 shared library. The newly created library produced by FTN95 is then used

in .NET projects or J2EE to implement the Grid services application logic.

4.2 Implementation of Grid Services

The workflow for a conventional Genetic algorithm is outlined in Figure 4(a). To implement it as a grid service, it is necessary to reengineer the workflow into the one illustrated in Figure 4(b). The new workflow involves requesting the evolved design variables for the subsequent generations after undergoing standard GA operating mechanisms (selection, crossover and mutation) and marshalling the fitness of the design points back to the Grid service. The main difference between these two workflows is exactly where the GA operators are applied.

```
gen = 1
Pop(gen) = randomly generated first generation
Evaluate fitness of all individuals in the population
While (termination condition = false)
gen = gen + 1;
apply genetic operators to Pop(gen)
evaluate fitness of the population
end
```

(a) Conventional GA

```
Get the initial generation from GA service
While (termination condition = false)
        Evaluate fitness of the whole population
        Notify the GA service with fitness values of the
        current population
        Get the next generation from GA service
End
```

(b) Re-engineered GA Grid Service

Figure 4: Logic for a Conventional Genetic Algorithm and as a Grid Service

The Grid service that delivers the Genetic algorithm maintains the control parameters of the algorithm together with the other configurations, and provides access to operations of the algorithm based on the standard SOAP protocol. Hence, our service for the OPTIONS genetic algorithm includes definitions of data fields that store the standard GA control parameters such as the population size, mutation rate and etc., as well as the following function calls that are made available:

- *ga_optdbs*, which initialise the internal data structure used by the algorithms;
- *ga_next*, which evolves the GA by one generation;
- *ga_objs*, which returns objective functions and constraints for the current population to the GA

In our approach, we choose to implement the GA as a collection of transient services, meaning that all the operations and data in the optimization process are handled by unique instances of the service that exist for the lifetime of the entire process. This provides a simple state management model, which avoids the need to transfer the state information back and forth between the service and client, or the need to deploy complicated mechanisms to manage states of multiple optimization processes. Details on how the service is implemented may be found in [21].

5. Service Consumption in the Matlab Scripting Environment

To consume the Web/Grid services, two client tools were developed in Java. Besides platform independence, the use of the Java programming language also enables seamless integration with Java-compatible scripting scientific environments, like Matlab and Jython, which are familiar to most engineers. One of the client tools provides access to the Genetic Algorithm services that drives the design optimization search process. The other is used to evaluate the high-fidelity airfoil analysis code in Singapore. The application example in the following section illustrates how the clients can be used with scripting languages to access these services to carry out airfoil optimizations. A piece of Matlab script used in the study is shown in Figure 5.

```
jobc = JobSubmissionClient;
ftpc = FileTransferClient;
gridftpc = GridFTPServiceClient;
% iterate through the GA process
npop=1;
for j=1:npop
  if j == 1 first=1;
  else first=0;
  end
  opt.ga_next(optdat, popsize, xvars,first, usepop);
  xvars=optdat.dwork;
  fid = fopen('designpoints.txt','w');
  for k1 = 1 : popsize
    for k2 = 1 : nvars
      fprintf(fid, '%g ', xvars(nvars*(k1-1)+k2));
    end
    fprintf(fid, '\n');
  end
  fclose(fid);
  % send input file via Web services
  inputfile = ftpc.sendFile('designpoints.txt');
  % remove the last newline character
  gridftpc.putfile('//tmp/',inputfile);
  jobid = jobc.submit(inputfile,1);
  while (jobc.jobpoll(jobid) == 'done')
    res = jobc.results(jobid);
    for i=1:popsize
      vars=xvars(nvars*(i-1)+1:nvars*i);
      xobj(i)=res;
      txobj((j-1)*popsize+i)=xobj(i);
      for k=1:ncons
        xcons((i-1)*ncons+k)=zeros(ncons,1);
      end
    end
  end
  opt.ga_objs(optdat, popsize, xvars, xcons, xobj);
end
```

Figure 5: Matlab script for the consuming of Web/Grid services

6. Experimental Studies and Discussions

Making use of the web and grid services provided at both sides of the institutional boundaries and a Matlab client environment, an optimization study carried out on a 2D airfoil aerodynamic design problem is presented next.

6.1. 2D Airfoil Aerodynamic Design

The 2D airfoil aerodynamic design optimization problem considered in this paper is an inverse pressure

design problem, which constitutes a good test case for validating the collaboration, as the target solution is known in advance. In the design problem, we choose the NACA 0015 as the target shape for a Mach value of 0.5 and 2-degree Angle of Attack (AOA), i.e. the desired pressure profile is computed using these conditions.

The airfoil is parameterised using the weighted sum of a number of basis functions, as shown in Equation (1). Here, the function series representation proposed by Hicks and Hennes [22] is used, which gives the upper (and lower) thickness y of the airfoil as the sum of the basis functions.

$$y = y_{basic} + \sum_{j=1}^{N} \alpha_j f_j(x) \qquad (1)$$

where x is the position along the chord, and α_j are design variables. The greater the number of parameters, the larger is the set of shapes represented, therefore the larger the design search space. In this work, 12 such functions have been employed, as shown in Figure 6. One such airfoil represented by series of Hicks-Henne functions is shown in Figure 7.

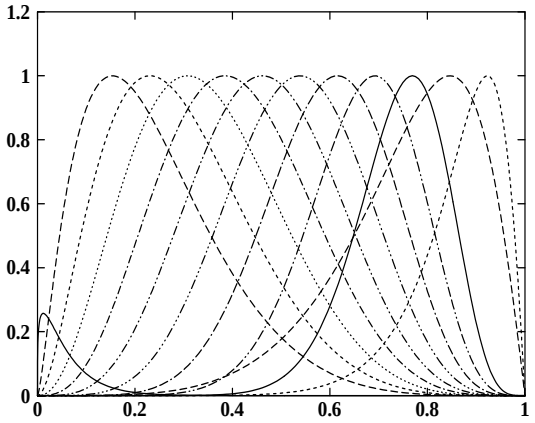


Figure 6: Airfoil Parameterisation using 12 Hicks-Hennes Functions

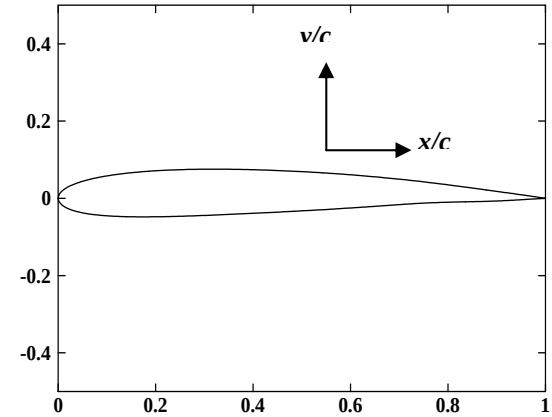


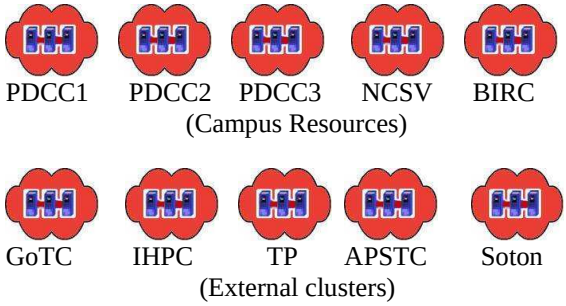
Figure 7: One of the Airfoils Represented by the Hicks-Hennes Functions

6.2. Grid Environment Setup

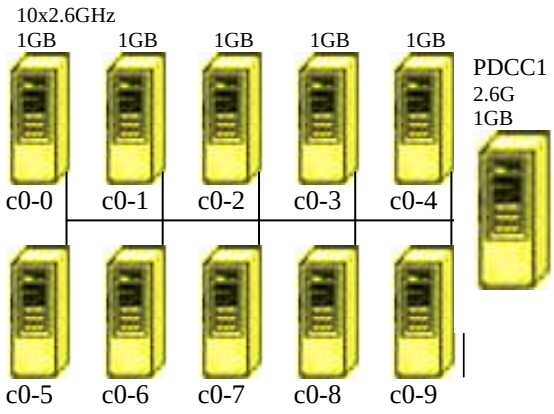
The Nanyang Campus Grid project is an initiative that links resources at different laboratories and research centre at NTU, including both national and international institutions to create a powerful Grid computing environment. The campus grid is made up of 10 clusters of heterogeneous platforms located at diverse geographical locations (see Figure 8a). This pool of grid resources varies from Sun Solaris to IBM AIX to Linux Itanium clusters. In this experimental study, we consider only one of the Linux clusters. The PDCC1 Linux cluster used in this study consists of 10 dual CPU compute nodes, and one master node, (see Figure 8b). The cluster configuration is Pentium IV Xeon 2.6GHz, with a total of 11 Gigabytes memory.

6.3. Result and Discussions

Here, a single exact adjoint airfoil code takes approximately 30 minutes to compute. When dealing with computationally expensive problems that cost more than a few minutes of CPU time per analysis or function evaluation, it makes perfect sense to compute the solutions in parallel using the available nodes on the campus grid. Using 24 design parameters, Figure 9 shows the result that converged to the known target solution design cycles successfully when using the grid service genetic algorithm, at Soton, and airfoil analysis web service deployed in NTU.



(a) 10 Clusters in the Nanyang Campus Grid



(b) PDCC1 Cluster

Figure 8: Nanyang Campus Grid

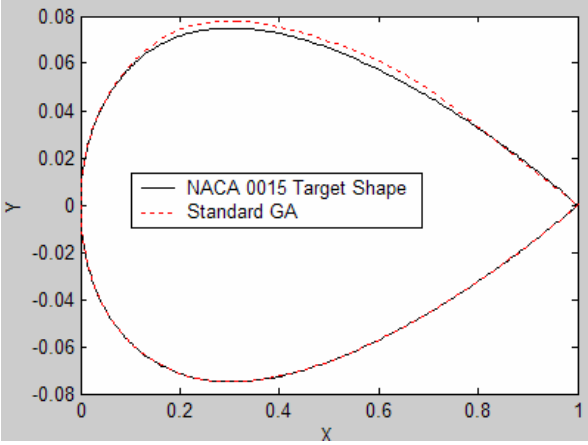


Figure 9: A plot on NACA 0015 target shape and final design using the grid service genetic algorithm at Soton, and airfoil analysis web service deployed in NTU.

From the preliminary experience gained from this collaboration, several observations may be made:

1) The Grid concepts and tools provide a workable alternative that enables collaboration across traditional institutional boundaries without jeopardizing individual intellectual property rights and software ownerships.
2) Proprietary programs on each side can be run on diverse platforms, using different programming languages, if both parties adopt the same SOAP interface.
3) A Service-oriented approach provides a seamless access to the sea of Grid and Web services and resources. This also accelerates research deliverables.
4) Last but not least, security and licensing issues needs to be better addressed if commercial codes are involved in the collaborations.

7. Conclusions

In this paper, the details of an ongoing research collaboration between School of Computer Engineering at Nanyang Technological University and Southampton e-Science centre at University of Southampton using a service-oriented approach for complex engineering design optimization has been presented and validated. Currently, a study of collaborative complex engineering design using a service-oriented approach has been successfully carried out. The present study was based on a single cluster in the Nanyang campus grid. Subsequently, in our future work, studies will be extended across multiple clusters within the campus grid.

Acknowledgement

The work at Southampton is supported by the UK e-Science Pilot project (UK EPSRC GR/R67705/01). The work at Singapore is supported by Nanyang Campus Grid. In addition, the authors would like to thank the Geodise team and all collaborators of the NTU campus grid and Dr. K. Y. Lum and Dr. Z. K. Zhang from Temasek Laboratories for contributing their airfoil analysis problem.

References

1. I.Foster, C.Kesselman, and S.Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations" , *International J.Supercomputer Applications*, vol. 15, no. 3, 2001
2. WS-Resource Framework, http://www.globus.org/wsrf/, 2004,
3. Simple Object Access Protocol (SOAP), http://www.w3.org/TR/soap/, 2004,
4. David E.Goldberg, "Genetic Algorithms in Search, Optimisation, and Machine Learning", 1989,
5. Web Services, http://www.w3.org/2002/ws/, 2004,
6. WSDL (Web Services Definition Language), http://www.w3.org/TR/wsdl, 2004,
7. The Globus Alliance, http://www.globus.org/, 2004,
8. Web Services Orchestration - a review of emerging technologies, tools, and standards, http://devresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf, 2003,
9. Nanyang Campus Grid, http://ntu-cg.ntu.edu.sg, 2004,
10. The Globus GSI, http://www-unix.globus.org/security/, 2004,
11. The Globus Resource Allocation Manager (GRAM), http://www-unix.globus.org/developer/resource-management.html, 2004,
12. GridFTP, http://www.globus.org/datagrid/gridftp.html, 2004,
13. Global Access to Secondary Storage (GASS), http://www.globus.org/gass/, 2004,
14. OPTIONS: Design Exploration System, http://www.soton.ac.uk/~ajk/options/welcome.html, 2004,
15. iSIGHT, http://www.engineous.com/index.htm, 2004,
16. ModelCenter: Integration for the engineering process, http://www.phoenix-int.com/products/ModelCenter.html, 2004,
17. Jython, http://www.jython.com, 2004,
18. Matlab, http://www.mathworks.com, 2004,
19. Compaq Visual Fortran 6.1, http://www.qtsoftware.de/dvf/dvf/v61.html, 2000,
20. FTN95 for Microsoft .NET and Win32, http://www.salfordsoftware.co.uk/compilers/ftn95/, 2004,
21. Song, W., Xue, G., Keane, A. J., and Cox, S. J., "Implementation of a Genetic Algorithm as a Grid Service", 2004, submitted to EuroPar 2004
22. Hicks, R. M. and Henne, P. A., "Wing Design by Numerical Optimisation" , *Journal of Aircraft*, vol. 15, no. 7, pp407-412, 1978