

# GROWL: A Lightweight Grid Services Toolkit and Applications

Mark Hayes and Lorna Morris  
University of Cambridge  
[mah1002@cam.ac.uk](mailto:mah1002@cam.ac.uk) and [ljm59@cam.ac.uk](mailto:ljm59@cam.ac.uk)

Rob Crouchley, Daniel Grose and Ties van Ark  
University of Lancaster  
[r.crouchley@lancaster.ac.uk](mailto:r.crouchley@lancaster.ac.uk), [d.grose@lancaster.ac.uk](mailto:d.grose@lancaster.ac.uk) and [t.vanark@lancaster.ac.uk](mailto:t.vanark@lancaster.ac.uk)

Rob Allan and **John Kewley**  
CCLRC Daresbury Laboratory  
[r.j.allan@dl.ac.uk](mailto:r.j.allan@dl.ac.uk) and [j.kewley@dl.ac.uk](mailto:j.kewley@dl.ac.uk)

## Abstract

As Grid technology matures and emerges from e-Science research as production systems such as the National Grid Service (NGS) and Campus Grids, its potential user-base will expand. Some of these user communities such as Bioinformatics and the Social Sciences rely on well-established software solutions involving R, Stata and Matlab. These applications do not easily fit into the typical grid environment. There is also a need for integrating heritage applications written in Fortran, C and C++ into the Grid in as seamless a way as possible.

To find an efficient solution for these communities, a prototype lightweight client-side toolkit called GROWL (Grid Resources on Workstation Library) was developed by CCLRC Daresbury Laboratory. Growl uses a client-server model to interface to existing Grid Services from applications written in Fortran, R, C and C++. GROWL is being further developed as part of the JISC-funded Virtual Research Environment (VRE) Programme by a collaboration from CCLRC Daresbury Laboratory and the Universities of Lancaster and Cambridge. The project will produce additional wrapper services (in particular for Stata and Matlab) and produce demonstrators in the fields of Bioinformatics, Computational Chemistry, Materials Science and the Social Sciences.

## 1 Background

The need for lightweight client toolkits was identified in late 2003. Chin and Coveney [1] listed a number of problems associated with Grid middleware packages then existing which were seen as a barrier to the uptake of Grid technologies, even in the more established computational sciences such as chemistry and physics:

*“We have encountered serious middleware-related problems which are hindering scientific progress with the Grid:*

- *The existing toolkits have an excessively heavy set of software and administrative requirements, even for relatively simple demands from applications;*
- *Existing toolkits are painful and difficult to install and maintain, due to excessive reliance on custom-patched libraries, poor package management, and a severe lack of documentation for end-users;*
- *Existing standards bodies and the task forces*

*within the UK e-Science programme are not engaging sufficiently with the applications community, and run a substantial risk of producing and implementing Grid architectures which are irrelevant to the requirements of application scientists.”*

They in turn cite Gabriel [2] who gives a cogent argument for the benefits for programming projects adopting a design philosophy of simplicity in cases where the right thing is too complex and conclude:

*“We argue that it is important to develop a simple, lightweight Grid middleware which is good enough for rapid adoption, rather than taking longer to develop a solution which will, supposedly, suit all needs. Such a toolkit must be:*

- *substantially more portable, lightweight, and modular in design;*
- *produced in very close collaboration with application scientists;*
- *sufficiently well-documented that end-users will be able to port existing codes to use Grid*

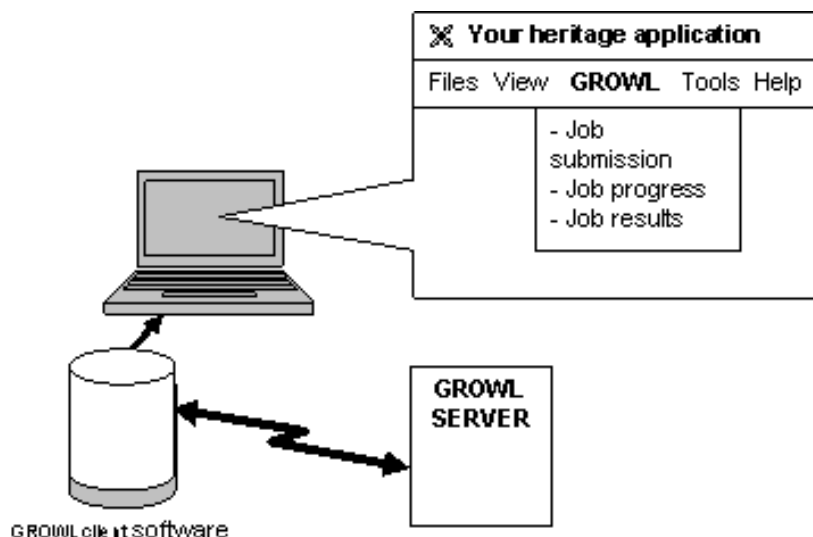


Figure 1: User's view of GROWL library

*techniques with the minimum of hassle."*

It was this that prompted the Grid Technology Group at CCLRC Daresbury Laboratory to produce the prototype GROWL (Grid Resources On Workstation Library [3]). It was based on the C/C++ language and used the gSOAP library to create a Web service client-server model in a library which interfaces to existing services on the Grid.

This work is now been carried forward by a JISC Virtual Research Environment project with collaborators at CCLRC Daresbury Laboratory and the Universities of Lancaster and Cambridge (for further information, see [www.growl.org.uk](http://www.growl.org.uk)).

The GROWL project aims to encourage the uptake of Grid-based computing and distributed data management, focusing on the issues which may hinder or facilitate end-user application development. Our solution builds upon the existing prototype GROWL library utilising Web Service technologies (in particular gSOAP). Particular requirements of the project are that GROWL be lightweight and extensible:

**Lightweight** implies that GROWL should be extremely easy to install, with functionality minimal but sufficient for the user requirements we have identified in target application areas. Likewise we will minimise any external dependencies, so for instance there will be no requirement to perform the installation as `root`; if the user already

has an appropriate version of gSOAP installed we will use it, otherwise we will download the latest version for them automatically. Additionally, only the parts of the library that are needed will be built and loaded into it.

**Extensible** means that it should be possible to easily extend GROWL to provide interfaces to additional middleware services or to use additional security mechanisms such as Shibboleth and PERMIS.

## 2 Web Service Approach

A Web Service is an application accessible using standard Internet protocols. Web Services represent black-box functionality that can be reused without concern for how that service is implemented, or what language it is written in and are accessed via ubiquitous Web protocols (e.g. HTTP) and data formats such as SOAP, WSDL and UML.

GROWL uses gSOAP to generate the client-side of the Web Services which are then wrapped by the GROWL API. The gSOAP compiler tools simplify the development of Web services by their provision of C and C++ language bindings for stub and skeleton code generation. This results in a flexible framework that can be also be utilised from Fortran. This would allow a subsequent version of GROWL to provide a Fortran library rather than provid-

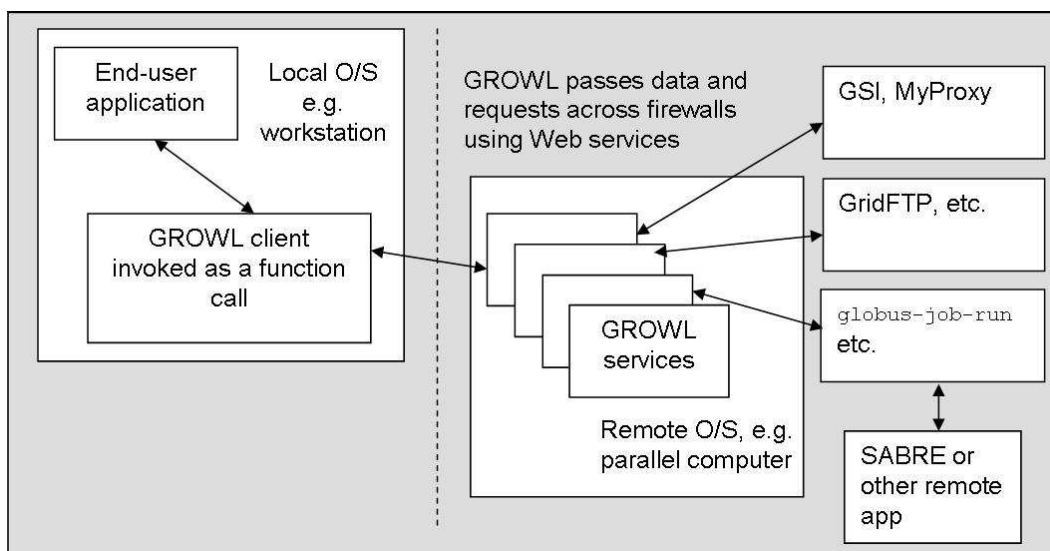


Figure 2: Web Service view of GROWL

ing Fortran wrappers to the GROWL C library. The gSOAP stub compiler automatically does all the data conversion from user-defined C and C++ data types to equivalent XML data types and vice-versa.

After selecting configuring which components of GROWL are needed, the user builds the GROWL library. Only the client-side parts of those requested services will be compiled and linked alongside the gSOAP generic client code. By taking this approach, GROWL takes the black-box one step further by embedding the client-side of the Web Service in the C library so that the API seen by the user has no mention of Web Services (as can be seen in Figure 1).

Once this is done, calls made to the GROWL library are redirected to the appropriate service on the GROWL Server and from there to an appropriate machine to execute the job (as is shown in Figure 2). It should be noted that since the GROWL Server is not over-constrained by institutional firewalls, it is accessible by submission and execute machines alike. This has the advantage that if the user's machine is unable to reach a given resource directly, the GROWL Server will act as a proxy for it. This, in a controlled way, circumvents the firewalls and means that the user can take advantage of Grid functionality without having to open any ports in his submission machine's firewall.

In order for the GROWL server to act as a proxy in this way, it must be able to act for the user. To ensure this is possible, GROWL provides an authentication service whereby users

can upload a proxy to their GSI certificate to a MyProxy server and can then instruct the GROWL server to request a delegated certificate to act on their behalf.

### 3 Application Areas

To demonstrate the general applicability of GROWL, along with the core generic services, demonstrators will be produced to support the Bioinformatics, e-Social Science and Computational Physics and Chemistry user communities. (These communities have differing requirements and the project is currently gathering these to ensure that GROWL has a coherent set of services):

#### 3.1 Bioinformatics

Microarray data analysis is notoriously resource intensive and requires the manipulation of large sets of data in a variety of different formats. A microarray is a glass slide, on to which DNA molecules (probes) are attached at fixed locations (spots). There may be tens of thousands of spots on a single array, and each spot contains DNA sequence from an individual gene. Microarrays are used for large scale analysis of gene expression, e.g. they can be used to compare the expression of thousands of genes in diseased and normal cells. Once the microarray data is collected it must be stored, compared with other data and analysed in order to make useful insights into the biological processes re-

sponsible for any observed changes in gene expression values.

Algorithms for the analysis of microarray data make use of a plethora of statistical methods, and the R [4] system is rapidly becoming the method of choice for this purpose. A collection of tools and extensions is being developed by statisticians and made available to the community via the Bioconductor project ([www.bioconductor.org](http://www.bioconductor.org)) to encourage this. MATLAB can also be used to implement complex statistical algorithms and it is particularly useful for matrix manipulation.

The GROWL strategy [5] is to enable ease of access to grid technology for biologists performing microarray data analysis using a variety of software tools, written in different languages (in particular R and C) through the use of the GROWL toolkit.

A variety of methods are used to organise gene expression data and look for patterns within the data set, typically clustering and class prediction algorithms. Markov chain Monte Carlo (MCMC) is a computationally intensive method for generating random samples from a distribution. The MCMC algorithm has been used to look for patterns in the microarray data, that can be used to predict tissue state. The output is a binary value for each gene, indicating whether the gene is a reliable indicator of tissue state or not. The algorithm has been implemented using MATLAB. To make this code suitable for running on a grid there are two options, to install MATLAB on the grid node, or to compile the code and generate an executable. In the second approach an accompanying archive containing the MATLAB component runtime files is also required.

To deploy the MATLAB executable the appropriate files need to be transferred to the external node and the paths to the library files set. A Condor submit script can be used to run the MCMC process on a node in an institutional condor pool.

GROWL will provide an alternative API and a variety of languages will be used to submit jobs from the client. The MCMC algorithm is easy to split up so that multiple jobs can be sent off simultaneously to different nodes, and the results of this sampling can then be combined, on the client side. GROWL will take care of finding an appropriate node to run the individual job and re-submitting if the job fails. The client will be able to monitor the progress of the jobs by polling. This same strategy can be employed for different types of application that can be split up into multiple independent

jobs and then notify the client when the jobs have completed so the client then can perform further functionality with the results.

Another application area is the design of DNA probes for microarrays. Identification of suitable probes requires calculation of the thermodynamic hybridization properties of candidate probes and assessing likely cross-reactions to non-targets from their DNA sequence. Present approaches are limited to traditional sequence similarity searches and rough approximations in calculating thermodynamic properties. Extensions made feasible by transparent access to distributed and specialized computing resources will allow DNA folding criteria to much earlier enter the scan for permissible probe candidates improving the quality of the obtained final set.

### 3.2 Chemistry and Physics

**Materials Science:** CCP3 is a Collaborative Computational Project (CCP) in the area of the computer simulation of surface structure and properties [6]. It aims to provide high quality, accurate and up to date computer codes for the analysis of a wide range of surface science experiments. The project is coordinated by the Materials Science Group at Daresbury Laboratory.

DL Visualize (DLV) [7] is a graphical user interface for use with a variety of materials simulation software including Crystal03 [8]. It provides visualisation capabilities and graphical interfaces to display and edit periodic structures in both two (surfaces) and three (crystals) dimensions.

DLV will use the GROWL library to make web service calls to the Grid on its behalf, utilising interfaces to stage executables, upload data, run jobs and download results. Apart from the acquiring and installation of a UK e-science certificate, the interface that the user sees should be unchanged.

Currently jobs can be submitted as computational jobs to local machines and some remote machines via `ssh`. The GROWL library will be used to Grid-enable DLV so that it can submit jobs to Grid resources using a single interface to various batch systems. Use will be made of GROWL services for file transfer (to stage executables and inputs, and return outputs), job submission and querying the status of a job.

**Computational Chemistry:** The e-CCP1 project is a collaboration between CCP1 (Quantum Chemistry) of the UK CCPs and CCLRC's

e-Science Centre [www.e-science.clrc.ac.uk](http://www.e-science.clrc.ac.uk). The project aims to develop, with broad consensus, mechanisms to facilitate the interoperation of quantum chemistry and other codes, allowing, for example, a plug-and-play approach to techniques such as QM/MM (Quantum Mechanics/Molecular Mechanics), or tools such as graphical user interfaces. The e-CCP project also builds on developments in Grid technology and other e-science activities to enable effective use of Grid resources by the quantum chemistry community.

The requirement is for a simple, lightweight set of tools that bridge the gap between the interfaces offered by Middleware, such as Globus, Condor and Storage Request Broker (SRB), that are being installed on Grid-enabled resources and the needs of the application programmer and/or advanced user who wants a number of compute tasks to be executed on remote Grid-enabled resources.

By using the GROWL library it will be able to develop an interoperable interface between the Grid and computational chemistry codes, such as GAMESS-UK [9] and Molpro [10]. The e-CCP1 project also builds on developments in Grid technology and other e-science activities to enable effective use of Grid resources by the quantum chemistry community.

### 3.3 Social Science

Although the typical size of the data collections used in Quantitative Social Science [11] are small compared to those encountered in Physics and Bioinformatics, the complexity of the statistical models required for analysis of the data is often computationally demanding. The need for model complexity arises because social data is often influenced by (amongst others) endogenous effect, shared random effects and clustering, missing data, dropout, and selection. Furthermore, the nature of survey data is often such that the explanatory variables are not independent. For example, when studying the effects of part time work and truancy on educational attainment it is not suitable to employ standard multivariate analysis since truancy and participating in part time work are potentially dependent.

Although most statistical software packages used in the social science community (notably Stata, R and S) can be employed to implement the appropriate models, the time necessary to undertake the analysis is usually prohibitively long. Several reasons account for this poor performance including the use of run time in-

terpreted code, executing software on low performance systems and the inability of the application software to exploit parallel architectures. However, there are a number of projects in which software has been developed that optimally implements certain models and in certain cases use MPI to run on parallel and clustered systems. An example of such a project is SABRE (Software for the Analysis of Binary Recurrent Events [12]). SABRE is designed to model recurrent events for a collection of individuals or cases and many other types of repeated measures data with binary, ordinal or count responses. It fits both standard models and various mixture models which allow for residual heterogeneity. It can be used to fit the following univariate statistical models

- binary data with logit, probit or complementary log-log link
- ordinal response data using a probit link
- count response data using a log-linear Poisson model
- continuous response using identity link

SABRE employs re-weighted least squares (for standard homogeneous models) and Newton-Raphson maximum likelihood (random effects binary models) algorithms. Both algorithms have been parallelized using Fortran MPI. The performance of a random effects logit model fitted in Stata using the `xtlogit` command with 12-point quadrature was compared to that of SABRE with both applications running on a single processor. In this study SABRE was found to be approximately 66 times faster. Furthermore the speed up of SABRE through parallelisation is almost linear. SABRE has now been extended to fit bivariate models and current work is undergoing to provide for n-variate analysis. The dimension of the space over which SABRE performs quadrature is equal to the number of variates, therefore the analysis time will increase geometrically with the number of variates modelled.

Because of the widespread requirement for the analysis of recurrent events, SABRE has been chosen as the application that will be used to demonstrate how GROWL services can be employed for e social science. Two key requirements for this part of the demonstrator are

1. Provide an interface for parallel SABRE in Stata, R and S (see Figure 3)

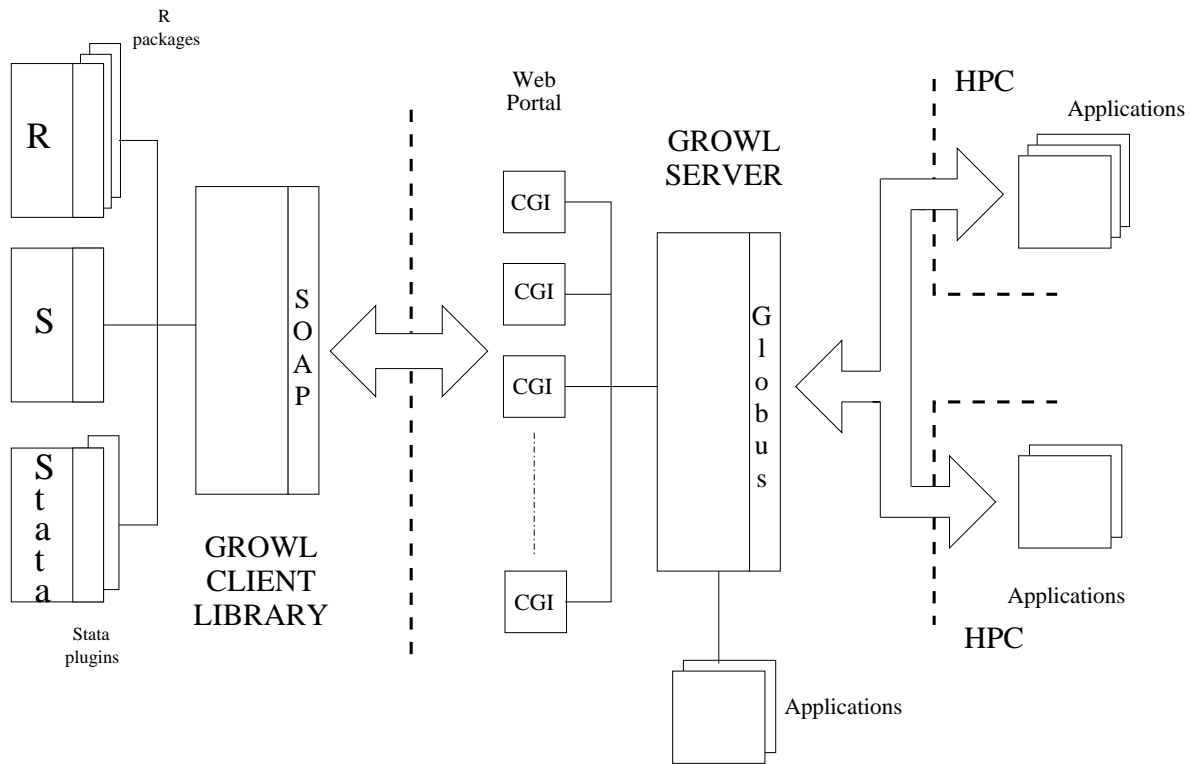


Figure 3: The use of language wrappers for GROWL

2. GROWL service to provide generic layer for communicating with existing applications

The first requirement is necessary because currently Stata, R and S are the statistical applications most widely used by social scientists. Furthermore, they are all easily extensible. The second requirement will allow parallel SABRE to be Grid-enabled using GROWL whilst obviating the need to modify existing SABRE code.

## 4 Related Work

WS-Grid [13], from the the University of Southampton, is a set of Java-based Web services that provide basic file and job submission services. Each node has an installation of Apache Axis and a subset of the WS-Grid services. Authentication is by username/password on each node. Users must have an account on each node.

WS-Grid is closely related to the OMII platform from the Open Middleware Infrastructure Institute [14]. OMII consists of a server installation and a lighter weight client tool, with services for job execution, file transfer, resource allocation and accounting. Authentication is by

X.509 certificate and SSL is used for transport layer encryption between server and clients. Authorisation is handled by “Process Based Access Control” (PBAC) whereby each Web service has a list of operations it can perform and PBAC determines which user can perform which operations in a given context. A “context” is a stateful exchange of messages between client and server. OMII comes with a small number of demonstrator applications in the form of services which reside in the server-side container.

WEDS (a WSRF-based Environment for Distributed Simulation [15] has emerged from the RealityGrid project [16] and the observations of Chin and Coveney. WEDS provides a mechanism for launching and interacting with computational simulations while they are running. WEDS is based on the WSRF::Lite [17]. Perl implementation of the Web Service Resource Framework[18]. It consists of four services representing

1. a CPU resource with information on specification, availability etc. and a method to invoke a new simulation process,
2. a wrapper around a running simulation process with methods for interacting with it (e.g. set new variables or return current

output),

3. file transfer, and
4. a broker which acts a registry for the above services and will match a simulation to an appropriate resource.

WEDS does not currently address authentication and authorisation across multiple administrative domains, although it intends to develop this capability using WS-Security.

## 5 Summary

There is clearly a need to make the use of Grid resources more attractive and tractable to end-user scientists. GROWL aims to reduce the learning curve by embedding the necessary technology into environments that are already familiar to our users.

GROWL takes a client-server approach based on Web services. Web services represent black-box functionality that can be re-used without worrying about how the service is implemented. GROWL takes this one step further by embedding the client-side of the Web service in a library linked to the original application, so that the interface exposed to the user has no mention of Web services. Calls made to the GROWL client library are redirected to the appropriate service on a GROWL server, which acts as a proxy to the remote Grid resource. Security credentials are delegated along this path using GSI.

The ongoing development of the toolkit is in close collaboration with end-user scientists from three application areas: computational chemistry, bioinformatics and social science. We will implement their applications as GROWL services, not only as useful examples, but also as a resource available to the whole community within each discipline.

### 5.1 Acknowledgments

We acknowledge funding for the work reported here from the JISC Virtual Research Environment (VRE) Programme [19].

## References

- [1] Jonathan Chin and Peter Coveney, “Towards tractable toolkits for the Grid: a plea for lightweight, usable middleware”, June 2004, [www.realitygrid.org/lgpaper21.pdf](http://www.realitygrid.org/lgpaper21.pdf)
- [2] R.P. Gabriel, “The Rise of ‘Worse is Better’” [www.jwz.org/doc/worse-is-better.html](http://www.jwz.org/doc/worse-is-better.html)
- [3] The GROWL Project, [www.growl.org.uk/](http://www.growl.org.uk/)
- [4] The R Project, [www.r-project.org/](http://www.r-project.org/)
- [5] Rob Crouchley, Ties van Ark, John Pritchard, John Kewley, Rob Allan, Mark Hayes and Lorna Morris, “Putting Social Science applications on the Grid”, First International Conference on e-Social Science, June 2005, Manchester, UK
- [6] Collaborative Computational Projects, [www.ccp.ac.uk/](http://www.ccp.ac.uk/)
- [7] DL Visualize, [www.cse.clrc.ac.uk/cmgl/DLV/](http://www.cse.clrc.ac.uk/cmgl/DLV/)
- [8] Crystal, [www.crystal.unito.it/](http://www.crystal.unito.it/)
- [9] GAMES-UK, [www.cfs.dl.ac.uk/](http://www.cfs.dl.ac.uk/)
- [10] Molpro, [www.molpro.net/](http://www.molpro.net/)
- [11] Centre for Quantitative e-Social Science (CQeSS), [e-social-science.lancs.ac.uk/cqess.html](http://e-social-science.lancs.ac.uk/cqess.html)
- [12] SABRE in R: [www.ncess.ac.uk/research/pdp/#ogsa](http://www.ncess.ac.uk/research/pdp/#ogsa)
- [13] WS-Grid, [dedalus.ecs.soton.ac.uk/WSGrid/](http://dedalus.ecs.soton.ac.uk/WSGrid/)
- [14] The Open Middleware Infrastructure Institute (OMII), [www.omii.ac.uk/](http://www.omii.ac.uk/)
- [15] Peter Coveney, Jamie Vicary, Jonathan Chin and Matt Harvey, “Introducing WEDS: a WSRF-based Environment for Distributed Simulation”, UKeS-2004-07, October 2004, [www.nesc.ac.uk/technical-papers/UKeS-2004-07.pdf](http://www.nesc.ac.uk/technical-papers/UKeS-2004-07.pdf)
- [16] Reality Grid, [www.realitygrid.org/](http://www.realitygrid.org/)
- [17] WSRF::Lite, [www.sve.man.ac.uk/Research/AtoZ/ILCT](http://www.sve.man.ac.uk/Research/AtoZ/ILCT)
- [18] WSRF, [www.globus.org/wsrf/](http://www.globus.org/wsrf/)
- [19] Virtual Research Environments Programme, [www.jisc.ac.uk/index.cfm?name=programme\\_vre](http://www.jisc.ac.uk/index.cfm?name=programme_vre)