JISC DEVELOPMENT PROGRAMMES
Project Document Cover Sheet

Final Report

Project

Project Acronym	GROWL	**Project ID**	None Assigned
Project Title	VRE Programming Toolkit and Applications (GROWL: Grid Resources on Workstation Library)		
Start Date	2005-02-01	**End Date**	2007-01-31
Lead Institution	University of Cambridge, e-Science Centre		
Project Director	Mark Hayes		
Project Manager & contact details	John Kewley, j.kewley@dl.ac.uk CCLRC e-Science Centre, Daresbury Laboratory, Keckwick Lane, Daresbury, Warrington, Cheshire, WA4 4AD tel: 01925 603513 fax: 01925 603634		
Partner Institutions	CCLRC Daresbury Laboratory, e-Science Centre University of Lancaster, e-Science Centre		
Project Web URL	http://www.growl.org.uk		
Programme Name	Virtual Research Environments (VRE) 5/04		
Programme Manager	Frederique Van Till Programme Manager (VRE) tel: 07875 338070 f.vantill@jisc.ac.uk		

Document

Document Title	GROWL: Final Report		
Reporting Period	Mar 2006 – Jul 2006		
Author(s) & Role	John Kewley (Project Manager), 01925 603513 Robert Allan Dan Grose		
Date	2007-03-31	**Filename**	GROWL_Final-1.pdf
URL	www.growl.org.uk		
Access	☐ Project and JISC internal		☒ General dissemination

Document History

Version	Date	Comments

Contents

1 Acknowledgements **4**

2 Executive Summary **4**

3 Background **5**

4 Aims and Objectives **6**

4.1 What Problems are we trying to address? 7

5 Methodology **9**

5.1 Technical Considerations 10

6 Implementation **11**

7 Outputs and Results **12**

7.1 NW-GRID Training 15

8 Outcomes **15**

8.1 Achievements against Objectives 16

8.2 Workshop on Lightweight Grid Computing 17

8.3 Comment on Success of Architecture 17

9 Conclusions **18**

10 Implications **19**

11 Recommendations **20**

1 Acknowledgements

Note the name of the JISC programme, and that the project was funded by JISC. The project may also want to list the project partners and acknowledge any person or organisation that was helpful during the project or in writing the report.

The GROWL project is grateful for the funding it received from the JISC Virtual Research Environment (VRE) Programme. It was a collaboration between the University of Cambridge e-Science Centre (lead institute), the University of Lancaster Centre for e-Science and the CCLRC e-Science Centre at Daresbury Laboratory.

The project was initially an 18 month project but received a no-cost extension to 24 months due to staffing problems. The project partners would like to thank the JISC for their support and flexibility especially with respect to this extension. We also acknowledge the support and comments we received from the Tavistock Institute during their VRE Programme evaluation.

Special thanks are also due to the various users who provided input to our User Requirements and tested the resulting software.

2 Executive Summary

Summarise highlights of the project (one page), including aims/objectives, overall approach, findings, achievements, and conclusions. The full report may include technical terms, but try to keep the executive summary in **plain English.**

Grid computing promises great things for Computational Scientists and Social Scientists. The reality is, however, that accessing computational jobs or data on clusters to the Grid from a desktop environment involves overcoming a variety of obstacles.

As Grid technology matures and emerges from e-Science research to deliver production systems like the National Grid Service (NGS) [17], the North West Grid (NW-GRID) [16] and Campus Grids [8], its potential user-base needs to expand. Several recent surveys have identified a need to reach out to user communities less familiar with the traditional supercomputing and large data centres. Effort at NeSC, the National e-Science Center in Edinburgh, and NCeSS, the National Centre for e-Social Science with its supporting hub in Manchester are investigating how to do this. Some of the “new” user communities such as Bioinformatics and the Social Sciences rely on well-established research procedures and software solutions such as R, Stata, SPSS and Matlab. These applications do not easily fit into the typical Grid environment, since they run on the user’s desktop computer rather than on a Grid resource. There is therefore a need for integrating “heritage” applications written in Fortran, C and C++ into the Grid in as seamless a way as possible to access the power of remote computing systems and access large datasets in a seamless way.

Our aims in this project were to encourage the uptake of Grid-based computing and distributed data management, focusing on the principal issues which could either hinder or facilitate application development to meet the requirements of end users. We refer to the difficulties identified as the “client problem” and suggested a solution to build upon the early prototype GROWL library to produce a truly lightweight extensible toolkit which complements other solutions.

From our initial requirements-gathering exercise and maybe even more so from our close inter-

working with practising quantitative and computational scientists, it has become clear that users' needs vary considerably. Contributing factors include whether they are coming from a background where cluster or parallel computing is the norm (in which case many aspects of the Grid paradigm are fairly accessible) such as in physics and chemistry. Other factors include which environments they are comfortable (or indeed uncomfortable) using, including the above-mentioned desktop applications but also bespoke GUI interfaces.

This led the development towards enhancing the original toolkit with the addition of the command-line GROWL Scripts for direct access to Grid resources and the development of a new GROWL server to support the SABRE-R project, making its functionality available as a demonstrator.

3 Background

Summarise the background to the project (and how it builds on previous work) and the need for it (and why its important).

The following quotation is from the CfP for the 8th IEEE/ACM International Conference on Grid Computing (Grid 2007) to be held in Austin, Texas, USA, September 2007 (`http://www.grid2007.org`).

Grid computing is evolving from the sharing of powerful computers for widely distributed applications to service orientation, open standards integration, collaboration, and virtualization. The ongoing development of the Grid as a service-oriented architecture for transparent and reliable distributed computing allows the reuse of existing components and information resources, and the assembly of these components in a flexible self-organizing manner. Hiding the complexity and technical details of the Grid from end users and application developers is going to play a crucial role in the future, for which issues related to security, provenance, automatic recovery, machine-interpretable metadata, QoS and SLA negotiation, etc. will have to be carefully addressed.

The need for lightweight Grid client toolkits was in fact 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 argues the benefits a simpler design philosophy in cases where the right thing is too complex, concluding:

"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 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 desktop applications to existing services on the Grid.

4 Aims and Objectives

List the aim and objectives agreed at the start of the project, and note if they changed during the project.

The aims of the GROWL project were 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. We proposed a solution that built upon an existing prototype GROWL library to produce a lightweight extensible Grid programming toolkit which complements other solutions, utilising Web Service technologies (in particular gSOAP). This can be used as the basis of a Service Oriented Architecture.

Lightweight implies that GROWL should be easy to install, with minimal but sufficient functionality for the user requirements we have identified in target application areas. It should also be possible to install the GROWL library quickly on a variety of client workstations running Linux or a similar UNIX-like operating system with a minimum of additional software.

Likewise any external dependencies should be minimised, so for instance there will be no requirement to perform the installation as `root` and only the parts of the library that are needed will be built and loaded.

Extensible means that it should be possible to easily extend the GROWL library to provide interfaces to additional middleware services (e.g. CONDOR, Netsolve, SRB) or to use additional security mechanisms (e.g. Shibboleth).

The project's intent was to fulfil these aims by meeting the following objectives:

1. Generate user requirements for a lightweight Grid application toolkit for the three target user communities outlined in the proposal;
2. Produce the GROWL toolkit with the condition that it should be possible to install GROWL quickly on a variety of client workstations running Linux or a similar UNIX-like o/s with a minimum of additional software;
3. Evaluate GROWL for Bioinformatics applications;
4. Evaluate GROWL for Physics and Chemistry applications;
5. Evaluate GROWL for Social Science statistical analysis applications.

4.1 What Problems are we trying to address?

This has been described in a number of presentations as the “Client Problem”. The notes below capture some of the discussion from the Workshop on Lightweight Grid Computing, see Section 8.2.

The idea of lightweight middleware has emerged to make the Grid more attractive to application developers for the following reasons:

- easy to use and install – no root access required and no special firewall ports open;
- provides transparent access to resources;
- small learning curve for application developers;
- easily wraps “heritage” high-performance computing applications, such as those from the UK Collaborative Computational Projects (CCPs) which have been developed over a period of 20 or more years.

In the context of a computational Grid infrastructure, we take “heavyweight” to mean software that is some combination of the following:

- complex to understand, configure or use;
- has a system “footprint” (disk space used, system resources used, etc.) that is disproportionately large when considered as a function of the frequency and extent to which the user uses the software;
- has extensive dependencies, particularly where those dependencies are unlikely to already be satisfied in the operating environment, or where the dependencies are themselves “heavyweight”;
- difficult or resource-intensive to install or deploy;
- difficult or resource-intensive to administer; and/or
- not very scalable.

We use the term “lightweight” to refer to software that is not “heavyweight”, i.e. it possesses none of the features listed above, or only possesses a very small number of them to a very limited extent. Thus lightweight software would be characterised by being:

- not too complex for end-users, developers and administrators to use and understand in their normal work context;
- easy to deploy in a scalable manner; and
- easy to administer and maintain.

It was noted that many of the items are “subjective”. Is “heavyweight” necessarily a negative thing? Not necessarily, but new users can get going more quickly with middleware that provides 80% of the full functionality but which is easy to install and use. AHE [5] and GROWL are examples of systems that are lightweight for the end users and solve the Grid client problem. The middle tier (server) is however not lightweight because it needs to have greater functionality. This is in common with portal solutions. The server is however hidden from end users, and is managed by an administrator and made to be reliable and secure. AHE and GROWL are also non-intrusive on resource providers, because they use whatever is already there, e.g. Globus or Condor. This makes the system usable for both providers and users – so we need to define for whom it is meant to be lightweight.

For adding a new application on the server, AHE is also lightweight as it involves editing two files. It does however have limited functionality and does not support multiple jobs yet. P-GRADE is similar. A wizard has been provided for AHE to add applications. GROWL uses C++ and is developing an automated “wrapping” capability.

It is reasonable for resource providers to install middleware to join the Grid and provide a commitment to offer a service. However probably only one kind of middleware will be installed on each resource. Considering the Web again, a Web browser is a complex application, but is packaged in such a way that it is easy to install. Likewise Web servers like Apache2 are now easy to install and support complex functionality. They even come packaged with O/S distributions. The Grid is not yet at this level of maturity.

It seems that we are really talking about usability in making these definitions. The underlying software might actually be quite complex.

There might be clear incentives for users to learn certain technologies, like a sophisticated text editor or a parallel computing environment. Users will use technologies if these give clear and obvious advantages. Is this a general statement? There are currently not many Grid users, and we need to make it usable so that more are willing to try and recognise the advantages. There is a danger (perhaps a long way in the future) that so many people take up the Grid that the resources are used up. However the Grid has access policies, e.g. to balance capacity vs. capability and it enables entirely new methods of working. We however do have to be careful that we are not just using the Grid instead of buying a bigger computer! Compute power is certainly not free and is only part of the story.

Nevertheless there are a number of technical issues to be addressed and this is what we have done in the GROWL project. The biggest issue is with institutional firewalls which inhibit the free transmission of data from one computer to another. In a Grid the large compute and data resources are known and professionally maintained systems and firewalls between them are allowed to be open. Users desktop client systems are however not treated in this way. Typically institutional servers, including Grid resources, are protected from un-trusted systems by a firewall. Even if a specific server has a number of open ports running pre-defined services, it is highly unlikely that any connection can be made through an institutional firewall back to a desktop client system. Typically desktop systems can only make outgoing socket calls, e.g. via port 80 or 8080 for HTTP or ports 22 or 443 for SSH or HTTPS. This is the basis of the “client problem” and implies that our lightweight software also needs to work over these ports by sending Web service commands and polling the servers for information.

5 Methodology

Summarise the overall approach taken and why this approach was chosen over other options considered. Then describe the methodology in more detail. Depending on the project, this might include the methodology for research you carried out, technical design or development, evaluation, etc. Finally, note any specific issues that had to be addressed by the methodology, e.g. standards, interoperability, scalability, etc.

There were important requirements in the GROWL project to enable access to "heritage" applications running on Grid resources and also to provide users with additional functionality via the environments with which they were already familiar.

The former is clear from the massive investment in the UK over more than 20 years in High Performance Computing (HPC) applications. Typical of this are the 12 or so Collaborative Computational Projects (CCPs).

The CCPs play a key role in the way computational science is coordinated in the UK. Each project brings together the major academic (and industrial) groups in a particular field to pool ideas and resources on software developments too large in scale, or too long-term, to be tackled successfully by any individual group. The CCP programme now includes more than 12 projects:

Collaborative Computational Projects	
CCP1	The Electronic Structure of Molecules
CCP2	Continuum states of Atoms and Molecules
CCP3	Computational Studies of Surfaces
CCP4	Protein Crystallography
CCP5	Computer Simulation of Condensed Phases
CCP6	Heavy Particle Dynamics
CCP7	Analysis of Astronomical Spectra
CCP9	Electronic Structure of Solids
CCP11	Biosequence and Structure analysis
CCP12	Parallel Computing in Fluid Dynamics and Numerical Modelling
CCP13	Fibre diffraction
CCP14	Powder and Single Crystal diffraction
CCPn	NMR Analysis
CCPB	Bio-informatics

For more information about the CCP Programme, see http://www.ccp.ac.uk.

The CCPs cover a wide range of computer applications in physics, chemistry and engineering, but are typical of the other emerging areas such as bio-informatics and quantitative social science which were considered in the GROWL project.

HPC computer applications are typically developed against advanced programming libraries, which provide functionality such as parallel message passing (MPI) and numerical algorithms (BLAS, SCALAPACK, NAG, FFTW, etc.) We intended GROWL to be acceptable to developers using a similar paradigm for Grid computing as a programming library which uses the GROWL API.

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 XML.

GROWL uses gSOAP to generate the client-side of the Web Services which are then wrapped by the GROWL API. The reason for this is explained in Section 5.1. 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 also be utilised from Fortran. This would allow a subsequent version of GROWL to provide a Fortran library rather than providing 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.

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 remote machine to execute the job (as is shown in Figure 1 below). 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 institutional or 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 (on behalf of) the user. To ensure this is possible, GROWL provides an authentication service whereby users can upload a proxy of their GSI (Grid Security Infrastructure) certificate to a MyProxy repository service and can then instruct the GROWL server to request a delegated certificate to act on their behalf.

GROWL functionality is therefore logically split into two main parts, the GROWL Client and the GROWL Server, corresponding to the client and server aspects of a Web Service. Note that some additional parts of the GROWL Client are not actually Web Service clients (e.g., the remote file handling routines) and these can only be used if there is direct access from the client machine to the target resource. Some other interfaces, e.g. to SRB the Storage Resource Broker, also do not use Web Services but require only outgoing ports to be open, which is usually the case.

The GROWL Client library is linked to the user's application and makes Web Service calls to the GROWL Server. The GROWL Server is a set of GROWL services running on a well known machine, accessible (with respect to firewalls) to the users application machine, the execution resources and a MyProxy Server. For each supported Web Service on the GROWL Server there will be analogous client functionality in the GROWL Library. Note that a GROWL client library should be able to talk to more than one GROWL Server for different GROWL services, as has indeed been tested.

5.1 Technical Considerations

Many of the applications discussed above are written in the Fortran language and have been adapted over many years to run on large parallel computers, typical of those available on Grids such as the National Grid Service and NW-GRID. Only a few, such as the CCP4 suite have been re-factored in other languages, in this case C++. They have user interfaces (in some cases GUIs) often written in C. For this reason it is important to use a middleware approach compatible with C and Fortran, which suggests Java is not appropriate. We additionally noted that the version of the Globus Toolkit, GT2.4.3, which is still widely used because of its maturity and stability as compared to newer versions (e.g. WSRF as implemented in GT4) is also primarily encoded in C with additional Bash and Perl scripts.

On the client side additional requirements were to expose the Web service calls provided by Globus in languages such as Perl, Python and R and to plug into desktop applications such as Stata, SPSS and MatLab in addition to the bespoke GUIs. Again Java was not required..

Our main dependencies in GROWL were therefore chosen to be as follows:

gSOAP `http://www.cs.fsu.edu/~engelen/soap.html` – Genivia Inc. had its own public open source license based on Mozilla license v1.1. and GPL. A commercial use license is also available. GROWL uses the gSOAP C/C++ library and Web services generator for its primary client-server interactions.

VDT – This software bundles several pieces of software, of which we make use of Globus, OpenSSH and MyProxy. The individual licences for these maybe viewed or downloaded from the VDT Web site (`http://vdt.cs.wisc.edu/licenses/` under the version number of VDT that is installed, 1.6.1 at the time of writing this document). They are also available after installation in `${GROWL_HOME}/vdt/licenses`.

Globus (source build) – For those Linux platforms not supported by VDT, we offer a "build from source" alternative. This downloads Globus, OpenSSH and MyProxy from their respective Web sites. These licenses are the same as described under VDT above.

SRB the Storage Resource Broker from San Diego Supercomputer Centre is widely used and has been provided with a simplified client interface in GROWL.

R is a powerful open-source language for statistical computing, used as a client environment popular in bio-informatics and the social sciences.

6 Implementation

Describe how you planned and implemented the project work and the activities it involved. Depending on the project, this might cover technical development, processes, how you conducted user studies, etc. Include any problems or issues that arose and how you handled them, where readers can learn from your experience. Tell the story of what you did rather than listing workpackages.

Each site was responsible for collating user requirements from their respective user communities. These requirements were collated into a User Requirements document. These requirements were converted into a proposal in an Architecture Document which was then discussed and agreed by the project developers and other stakeholders after suitable revision. This resulted in a three-tier architecture as outlined above.

Figure 1 shows, from left to right: (1) possible client environments which end users might interact with; (2) the GROWL Client Library with generic functions able to plug into the various desktop environments; (3) GROWL typically makes Web service calls across a possible firewall (illustrated by the dotted line) to a server; (4) the server(s) contains a set of Web services illustrated as CGI processes which are invoked by a dispatcher when a service request is de-serialised; (5) the service

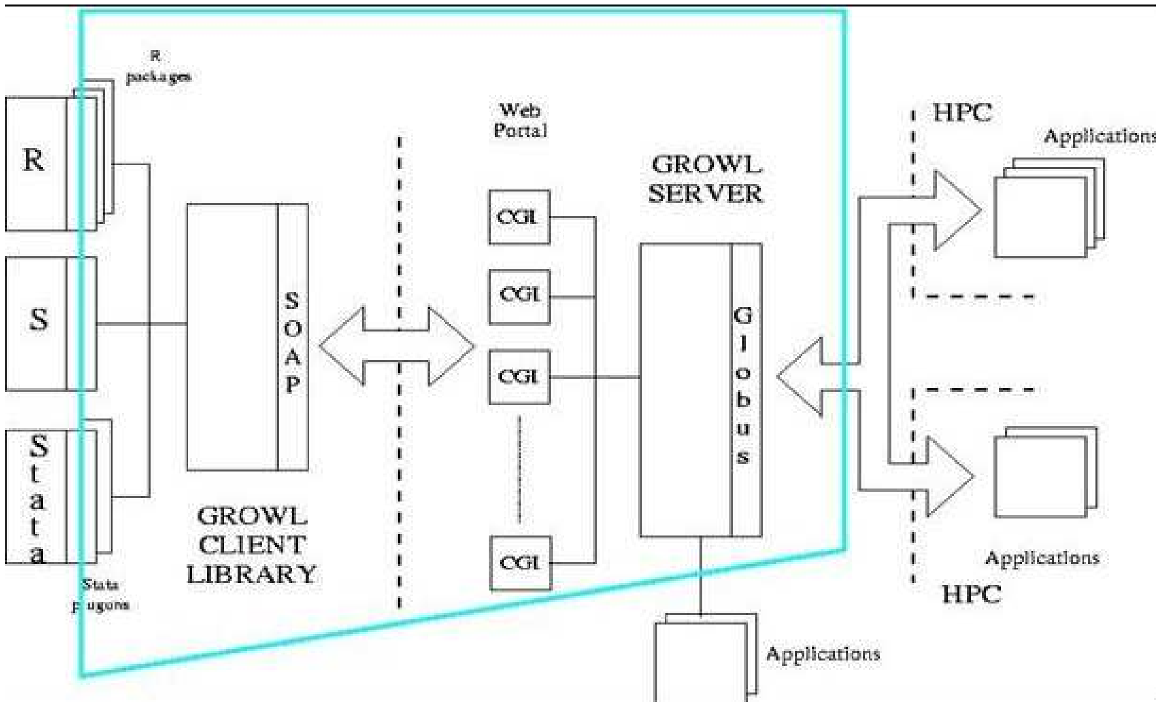

Figure 1: GROWL Architecture

invoked may wrap more complex functionality, such as Globus Grid middleware as illustrated; (6) this transfers data and invokes applications on a remote HPC server. In this figure, the area inside the large turquoise trapesium is to be addressed by developers in the GROWL project, the clients on the right are used by the end users and the HPC applications and services on the right are installed on Grid resources and the responsibility of system administrators. We will return to this distinction in Section 8 below.

In the initial stages of implementation of the project, Lancaster were responsible for enhancements to the GROWL Server and Daresbury for the GROWL Client with Cambridge taking a lead with early user adoption. Each site then investigated different server and client components and how these met the requirements of their respective user communities.

During the course of our preparatory work two additional courses of action emerged from analysis of the user requirements: (1) an enhanced server with the capability to "wrap" applications and act as a proxy to them and to handle multiple user "sessions"; (2) a command-line interface enhancing a set of "helper scripts". This is further discussed in Section 7.

7 Outputs and Results

Explain the end result of the project work in an objective way. Depending on the project, it might include research results, findings, evaluation results, data, etc. If the project created something tangible like content, a portal, or software, describe it. Engage the reader, and avoid a long list of *deliverables.*

Based on initial requirements and feedback received during the project, the GROWL developers evaluated three different but closely related approaches to delivering the power of Grids to users' desktop environments. Each approach has positive and negative features, but together form a comprehensive toolkit which could be extended to meet a greater range of applications. In work on real applications a combination of approaches is typically used. We have not had time to systematically analyse the spread of usage by different communities.

The three approaches adopted are explained in the table below.

Approach	Benefits	Disadvantages
Script interface	simplifies installation on desktop; easy wrapping for Perl, Python and R; direct access to Grid resources; simpler than Globus commands; can provide a unified abstract shell interface; provides "helper scripts" for handling certificates etc.	may require firewall ports to be open; relatively large "footprint" of dependencies
Generic Web service library	can be linked into "heritage" applications which have plug-in capability, e.g. Stata, SPSS, Matlab; can be linked into bespoke GUI and other desktop applications; can invoke remote services if their location and name is known; can be wrapped to bind with other languages	may be too generic; aimed at developers rather than end users; work has to be done to specify the task to be invoked
Application-specific services	provides bespoke client for complex stateful applications via the GROWL server; client API reflects the tasks required; client aimed at end users but limited to pre-defined functionality	wrapping technology is complex and requires further work, but promising; definitely aimed at developers

Several areas of functionality were targeted as requested by users before and during the GROWL development. These are listed below and typically each one has been implemented as a "module" within the GROWL toolkit.

Install software: scripts to install required software on the user's desktop machine. Principally aimed at Linux users, we have not had available effort to do this on WIndows platforms (CygWin can be used or a shell onto a Linux platform).

Authentication: manage certificate formats, "log on" to the Grid, check status, log out. This uses GSI, the Grid Security Infrastructure. It does not yet use Shibboleth.

Data Management 1: SRB (the Storage Resource Broker from San Diego Supercomputer Centre) is available via a simplified client interface. This enables management of files or "collections" locally and on remote SRB servers.

File Management 1: upload and download files from desktop to the GROWL server or to a remote Grid resource (latter subject to firewalls). This uses either data included with the SOAP messages or GSI-SCP.

File Management 2: use GridFTP to move files between Grid resources. This has not been fully tested, but would provide enhanced performance when necessary.

Job Management 1: submit jobs to batch queues or interactively on a remote Grid resource currently via Globus. User can check status, retrieve output, clean up.

Workflow Management 1: A simple workflow tool, Schedule, has been included wih GROWL but not used in anger yet.

A number of tasks are still to do, and these form part of the ongoing work of deploying GROWL for use in other research projects:

Metadata management: the Rcommands developed in the NERC-funded e-Minerals project are soon to be added to GROWL.

Data management 2: OGSA-DAI will be integrated with GROWL via a simplified Web service interface. This will enable access to national data archives as in the JISC-funded GEMS project which bridges from the NGS to UKDA.

Job Management 2: A job submission interface using Condor or Condog-G is to be developed based on work already carried out at Daresbury and University of Cambridge.

Workflow Management 2: The RMCS software developed in the NERC-funded e-Minerals project has a Web services interface to control workflow usage scenarios for parametric studies of many different phenomena. This will be integrated with GROWL in future work for rollout to, for instance the Diamond e-Science Infrastructure.

Both scripts and C API have been used to implement many of these requirements. We note that it is possible to install GROWL on both client and server, enabling one GROWL module to call another and thus create a “GROWL Grid”.

Interfaces have been tested as follows for a number of applications:

Helper scripts: used for tasks such as authentication.

Bash scripts: augment the helper scripts and used to directly manage Grid resources where possible and can be effective for experienced users.

C programming library: demonstrated in sample C applications for physics and chemistry.

R interface: wraps the C library to provide an alternative scripting interface in the R functional language. Used by social scientists and bio-informaticians. It is possible to automatically create the R functions from the corresponding C client-side functions.

Plug-in interface: uses the C library to access GROWL functionality from applications with plug-in technology such as MatLab and Stata. This has not yet been fully tested.

The R interfaces has been used by bio-informaticians and also social scientists and there is significant interest from researchers at other institutions.

We note that various components of GROWL can be downloaded from the project Web site[3]. This will continue to be maintained for the foreseeable future.

7.1 NW-GRID Training

Project members were instrumental in the running of the first NW-GRID training event. The goals of this course were to give participants:

- An understanding of the concepts of Grid Computing and e-Research
- An orientation to the NGS and NW-GRID and how to access them
- Sufficient practical experience to allow initial use of NW-GRID services in future

The enabled the software outlined above to be introduced to a larger number of potential users looking for easy access to Grid resources. GROWL Scripts were used to show delegates how to install Grid middleware, transform their certificates and run Grid jobs with comparative ease.

8 Outcomes

In this section, assess the value of the project work. List project achievements against the aims and objectives set. Summarise project outcomes and their impact on the teaching, learning, or research communities. Indicate who will benefit from the work, how, and why. Also comment on what you learned that may be applicable to other projects, e.g. whether the methodology worked.

During the course of the project three classes of user were identified. This was presented at the 1st International Conference on e-Social Science, Manchester June 2005 and also at the JISC Joint Programmes Conference that year. It uses an adaptation of the “Classification of Grid Users” of Foster and Kesselman:

Class of User	Purpose	Requires	Concerns
End users (e.g. quantitative social scientists).	Do research. Solve problems.	Applications.	Transparency, ease of use, performance.
Application developers.	Develop new and extend existing applications.	Programming tools, APIs, libraries.	Ease of use, reusability.
Tool developers.	Develop APIs, toolkits, libraries.	Grid services.	Adaptivity, applicability, robustness, stability.
Grid developers.	Provide Grid services.	Local system services.	Security, connectivity, protocols.
System administrators.	Manage Grid resources.	System tools.	Balancing local and global concerns.

GROWL has been designed to target the Application Developers and the Tool Developers and is a toolkit to write Grid-enabled applications. It uses standard middleware from Grid Developers so addresses issues of security, connectivity and protocols and it enables development of rich Grid-enabled applications to satisfy the requirements of End Users in carrying out their research.

8.1 Achievements against Objectives

1. *"Generate user requirements for a lightweight Grid application toolkit for the three target user communities."*

   A User Requirements document has been produced following meetings with a variety of potential users, both individually and at workshops.

   The "Collaboration for Quantitative e-Social Science (CQeSS)" project has expressed an interest in using GROWL for their work and Adam Braimah (DL) has produced an SRB client for use with GROWL. He is currently investigating whether OGSA-DAI can be handled by GROWL.

   An interface to the GROWL SABRE service hosted at Lancaster is being provided as a free plug-in for the commercial statistics package Stata. This development is being undertaken in recognition of the need to make the Sabre service running on a Grid hose available from within the software tools most commonly employed by the social science and economics research communities.

   Since several CCLRC staff have been using GROWL scripts as an easy way to install the VDT Globus tools, this work will be spun-off as a separate part of the GROWL client since it can be used independently of the Server and is being trialled for access to the NGS, NW-GRID and SCARF resources.

2. *"Produce GROWL toolkit: it should be possible to install the GROWL library quickly on a variety of client workstations running Linux or a similar UNIX-like o/s with a minimum of additional software."*

   The following groups in Computational Science and Engineering (CSE) at Daresbury Laboratory (in addition to those mentioned under Physics and Chemistry below) are using GROWL as an easy way to install VDT and for certificate transformation.

   - Computational Engineering Group (Simon Mizzi)
   - Band Theory Group (Martin Lüders)
   - Advanced Research Computing Group (Martin Plummer)

3. *"Evaluate GROWL for Physics and Chemistry applications."* The following groups are using GROWL within the CSE Department at Daresbury Laboratory:

   - Quantum Chemistry Group: Jens Thomas has used GROWL scripts as a Globus client interface for their CCP1 GUI [14]. This provides a graphical interface for community codes including GAMESS-UK [15].
   - Materials Science Group, Barry Searle is using the GROWL C interface to submit jobs to Grid resources from DL Visualize [10].

4. *"Evaluate GROWL for Social Science statistical analysis applications."*

   - The GROWL C API is used in conjunction with an existing R package (sabreR) [4] to provide an interface to parallel SABRE (Software for the Analsyis of Binary Recurrent Events) [12]. This work is being undertaken by Lancaster University and Daresbury Laboratories and is funded by CQeSS and NW-GRID.

- The GROWL server is being used to access a grid hosted *k Nearest Neighbours* (kNN) algorithm. The resulting service is being employed to extend an existing R package (`spgwr`) [13] which is used for Geographically Weighted Regresison (GWR). This work is being undertaken by Bristol and Lancaster Universities and is funded by the ESRC.

5. *"Evaluate GROWL for Bioinformatics applications."*

   - GROWL was used for some Markov Chain Monte Carlo simulations by Peter Sykacek of the Bioinformatics and Genomics Group (Micklem Lab) in the University of Cambridge Department of Genetics, unfortunately he took a new position before this could be fully evaluated.
   - An R interface for accessing the GROWL Scripts functionality has recently been produced for future evaluation by this community.

8.2 Workshop on Lightweight Grid Computing

Project members (with Peter Coveney of UCL) organised the Workshop on Lightweight Grid Computing in Hope, Derbyshire in May 2006 to discuss and share ideas on how we can give scientists simpler tools to utilise the Grid more effectively. A full report of the event is in preparation. The objectives of the workshop were:

- To share ideas on the middleware needed to provide users with lightweight access to Grid Resources
- To look at ways to pool resources for future collaboration *(note that the use of portals to provide this capability was not considered since the workshop focused on programmatic and scripted interfaces to the Grid)*.

The event was considered a success and presentations are available from the event Web site [18].

A short report entitled *Lightweight Grid Computing and Usability* was submitted as input to the NeSC workshop on "User Requirements and Web-based Access for Grid Research", NeSC, 2006-05-19 (report available from GROWL Web site[3]).

8.3 Comment on Success of Architecture

To achieve maintainable, stable, lightweight, platform independent client interfaces for accessing Grid resources it is necessary to reduce, as far as is practically possible, the amount of Grid logic contained within the client system. This due to a number of issues including:

- Grid middleware is not yet stable which leads to continued modifications to the implementation of Grid based solutions. Thus, to provide stable and maintainable client solutions it is necessary to isolate the implementation from the client interface.

- Different clients will have access to different Grid resources offering the same functionality. Thus, unless the Grid logic is separated from the client, it will have to account for all vagaries of a plethora of different interfaces to essentially the same Grid application.
- When a client has access to more than one Grid compute resource it is often necessary to decide how a job can be subdivided across these multiple resources. Since the Grid resources are not well interconnected between themselves it is necessary for decisions regarding job scheduling to take place elsewhere. Currently this is in the client system itself, which is unacceptable for the foregoing reasons.

It is well known that an appropriate solution for isolating implementation from client and service provider is a three-tier architecture. In addition, such an architecture serves to minimize the amount of information each resource requires about other resources and reduces the number of communication pathways. This prevents the growth of an unmanageable "stove pipe" solution that is non scalable. However, given the large number of Grid based applications and the extent of the development community required to develop them, such an architecture needs to be provided *a priori* to the application development itself. Thus, in the first instance, it is necessary to develop three-tier frameworks with provide easy to use, open APIs for the developer community. Such technologies already exist, such as CORBA, web services and many others. However, none of these, on an individual basis, fully take into account the following requirements which appear fundamental to a three-tier Grid architecture:

- Multiple service implementations provided through a common interface where the interface reflects the context of the problem domain, not the specifics of the Grid middleware;
- Secure communication;
- Client identification;
- Client lifetime is different from service lifetime, thus services have to be stateful to be able to efficiently and effectively manage job creation, submission, retrieval and lifetime;
- Clients can only communicate through a limited number of ports;
- Common format for communication with mappings to a number of different languages.

It is considered that provision of such a three-tier architectural framework offering the above facilities is achievable. That this is so is supported by positive outcomes from preliminary pilot studies undertaken in existing projects for example, GROWL, AHE, RMCS and GridSAM. These projects were compared and contrasted at the Workshop on Lightweight Grid Computing and could be combined in future work to form a more comprehensive toolkit.

9 Conclusions

Briefly summarise any conclusions that can be drawn from the project work.

Providing a common framework for Grid developers to work with is vital to enable a large, disparate development effort, spread across numerous research groups to be able to provide stable, maintainable lightweight client interfaces to Grid resources. Developer tools and APIs need to be produced in advance of, or at least in parallel with, the development of client applications. It is proposed that the preliminary development undertaken as part of the GROWL project is continued and extended, with reference to other projects with similar objectives, to achieve this goal.

Our work has resulted in:

- A good "demonstrator" that is being actively used, although in need of "hardening"
- The GROWL Server
- The GROWL Scripts which are in widespread use, helping computational scientists run their jobs on the Grid.

10 Implications

Consider the future implications of your work and how others can build on it. What are the implications for other professionals in the field, for users, or for the community? What new development work could be undertaken to build on your work or carry it further?

Need for Grid middleware developers to consider the client-side problem, whether firewalls, certificates or sheer amount of s/w to download, even VDT client.

Following our suggestions to make Grid Software easier to use, the following *changes have been accepted* by our 3rd party software suppliers:

gSOAP It is typical when gSOAP is used that either a client or a server is built, but not often both together. We requested that command-line options were provided for `soap2cpp` to produce client- or server-only stubs. The command-line options (`-C` and `-S`) have now been added to gSOAP for this purpose

MyProxy When the MyProxy command `myproxy-init` is used to upload a medium term proxy to a MyProxy server, it is frequently the case that the user will want to generate a local proxy immediately afterwards. At our request, the MyProxy development team have added the `-l` option to `myproxy-init` to support this functionality.

GSI-enabled OpenSSL The ANL staff have enhanced their Globus certificate handling commands to permit generation of proxies directly from the PKCS#12 (`.p12` or `.pfxA`) format certificates rather than relying of conversion to `.pem` format. MyProxy have also updated their code and we await a final bug to be fixed before this functionality can be fully utilised in the GROWL Scripts.

VDT When a client only download of VDT is requested, the amount of software that is additionally installed is considerable. The VDT team have agreed this is a dependency problem and they will look into it at some point.

Condor Members of the Condor team agreed the benefit of embedding GROWL Scripts functionality in the Condor-G submission back-end

11 Recommendations

Optional: List any specific recommendations for the teaching, learning, or research communities.

We recommend combining existing technologies such as GROWL, AHE, GridSam and RMCS to form a single comprehensive Grid-enabled VRE Programming Toolkit for the whole research community. Currently no funding is available for this work however and the projects will carry on separately supporting a fragmented community.

List any references to the work of others you have cited (e.g. articles, reports, studies, standards), and any explanatory notes. Provide URLs for any materials available on the web.

References

[1] Jonathan Chin and Peter Coveney, *“Towards tractable toolkits for the Grid: a plea for lightweight, usable middleware”*, June 2004,
http://www.realitygrid.org/lgpaper21.pdf

[2] R.P. Gabriel, *“The Rise of ‘Worse is Better”*
http://www.jwz.org/doc/worse-is-better.html

[3] The GROWL Project,
http://www.growl.org.uk/

[4] D.J. Grose et al, *“SabreR - Grid-Enabling the Analysis of Multi-Process Random Effects Data in R”*, NCeSS Conference, June 2006.
http://www.ncess.ac.uk/events/conference/2006/papers/abstracts/GroseSabreR.shtml

[5] P.V. Coveney, R.S. Saksena, S.J. Zasada, M. McKeown and S. Pickles The Application Hosting Environment: Lightweight Middleware for Grid-Based Computational Science. Computational Physics Communications (1007) in press

[6] The R Project,
http://www.r-project.org/

[7] Nearest Neighbour

[8] UK CampusGrid meeting,
http://www.nesc.ac.uk/esi/events/556/

[9] Collaborative Computational Projects,
http://www.ccp.ac.uk/

[10] DL Visualize,
http://www.cse.clrc.ac.uk/cmg/DLV/

[11] Centre for Quantitative e-Social Science (CQeSS),
http://e-social-science.lancs.ac.uk/cqess.html

[12] SABRE,
http://www.lancs.ac.uk/staff/cpajp/sabre/

[13] SPGWR,
http://cran.r-project.org/src/contrib/Descriptions/spgwr.html

[14] CCP1GUI,
http://www.cse.clrc.ac.uk/qcg/ccp1gui/index.shtml

[15] GAMESS-UK,
http://www.cfs.dl.ac.uk/gamess-uk/index.shtml

[16] NW-GRID,
http://www.nw-grid.ac.uk/

[17] NGS,
http://www.ngs.ac.uk/sites/

[18] Lightweight Grid Workshop,
http://tyne.dl.ac.uk/GROWL/Lightweight.shtml

[19] Department of Genetics, Bioinformatics and Genomics Group (Micklem Lab),
http://www.gen.cam.ac.uk/Research/micklem.htm

[20] Peter Coveney, Jamie Vicary, Jonathan Chin and Matt Harvey, *"Introducing WEDS: a WSRF-based Environment for Distributed Simulation"*, UKeS-2004-07, October 2004,
http://www.nesc.ac.uk/technical_papers/UKeS-2004-07.pdf