The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)______________________________
Summary

The objectives of this project consist of three areas. The first area is a general research into the world of Grid to gain understanding of the basics of Grid computing. The second area is the specific research into Grid resource management and scheduling, including the study of Grid Resource Brokers and Grid portals. Details of Nimrod/G is also examined and documented. The third area of this project is to design and implement a Grid portal for Nimrod/G. At the end of implementation the prototype portal is tested and evaluated and the results are documented in this report.

At the conclusion this project has met all the objectives set out at the start. The project has met all the objectives laid out above and in some areas has gone above and beyond these objectives. Not only has the project met the objectives set out, it has been a valuable learning experience for me, further details of how the project has met its objectives etc can be found in the evaluation and conclusion chapters of this report.
Acknowledgements

There are many people that I wish to thank for their help and assistance in this project.

First I want to thank my supervisor Dr. Karim Djemame, who gave me a lot of advice for the design of the portal and the structure of the report and guided me through the whole process of the project.

Thanks to Professor David Abramson and his Nimrod team, without whose warm-hearted replies the design of the portal would not be solved.

Also thanks to my dear classmate friends, who have always been encouraging and willing to help, especially when I am feeling stressed.
**List of Acronyms**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CGI</td>
<td>Common Gateway Interface</td>
</tr>
<tr>
<td>GIIS</td>
<td>Grid Index Information Service</td>
</tr>
<tr>
<td>GPDK</td>
<td>Grid Portal Development Kit</td>
</tr>
<tr>
<td>GRB</td>
<td>Grid resource brokers</td>
</tr>
<tr>
<td>GridFTP</td>
<td>Grid File Transfer Protocol</td>
</tr>
<tr>
<td>GRIS</td>
<td>Grid Resource Information Service</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>JavaCoG Kit</td>
<td>Java Commodity Grid Kit</td>
</tr>
<tr>
<td>JSP</td>
<td>Java Server Page</td>
</tr>
<tr>
<td>LSF</td>
<td>Load Sharing Facility</td>
</tr>
<tr>
<td>MVC</td>
<td>Model-View-Control</td>
</tr>
<tr>
<td>OGSA</td>
<td>Open Grid Service Architecture</td>
</tr>
<tr>
<td>PBS</td>
<td>Portable Batch System</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>SGE</td>
<td>Sun Grid Engine</td>
</tr>
<tr>
<td>TCP</td>
<td>Transfer Control Protocol</td>
</tr>
<tr>
<td>TFE</td>
<td>Task Farming Engine</td>
</tr>
<tr>
<td>WAN</td>
<td>Wide Area Network</td>
</tr>
<tr>
<td>WRG</td>
<td>White Rose Grid</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Table of Contents

Summary ................................................................. I
Acknowledgements ................................................... II
List of Acronyms ....................................................... III
Table of Contents .................................................... IV
Table of Figures ...................................................... VII

Chapter 1. Introduction .............................................. 1
1.1 Project Aim ....................................................... 1
1.2 Project Objectives .............................................. 1
  1.2.1 Understand Grid Computing ......................... 1
  1.2.2 Understand Grid Resource Management and Scheduling ....... 1
  1.2.3 Study Nimrod/G ........................................... 1
  1.2.4 Design a portal for Nimrod/G .......................... 2
  1.2.5 Implement the portal on grid test bed ................. 2
  1.2.6 Integrate the Portal with Nimrod/G ................. 2

Chapter 2. Grid Computing and Globus Toolkit ............. 3
2.1 Introduction ....................................................... 3
2.2 The Grid .......................................................... 3
  2.2.1 Security .................................................... 3
  2.2.2 Resource Management and Scheduling .............. 4
  2.2.3 Information Service ...................................... 4
  2.2.4 Data Management ....................................... 4
2.3 Globus Toolkit ................................................. 4
  2.3.1 Grid Security Infrastructure (GSI) ................. 5
  2.3.2 Grid Resource Allocation Manager (GRAM) ....... 5
  2.3.3 Monitoring and Discovery Service (MDS) .......... 6
  2.3.4 Grid File Transfer Protocol (GFTP) ................ 6
2.4 Future development of the Grid ......................... 6
2.5 Summary ....................................................... 6

Chapter 3. Grid Resource Management & Scheduling .... 7
3.1 Introduction ...................................................... 7
3.2 Grid Resource Brokers ....................................... 7
  3.2.1 Sun Grid Engine (SGE) ................................. 7
  3.2.2 Condor-G ................................................ 7
  3.2.3 Portable Batch System (PBS) ......................... 7
  3.2.4 Load Sharing Facility (LSF) ......................... 8
3.3 Nimrod/G ....................................................... 8
  3.3.1 Parametric Language ................................. 9
  3.3.2 Computational Economy ............................. 9
  3.3.3 Nimrod/G Clients ..................................... 9
  3.3.4 Nimrod/G APIs ...................................... 9
3.4 Grid Portals ................................................... 9
  3.4.1 HotPage User Portal .................................. 10
  3.4.2 White Rose Grid (WRG) Portal .................... 10
  3.4.3 Nimrod Portal .......................................... 10
3.5 Summary ....................................................... 10
Chapter 4. Design of the Portal................................................................. 11
  4.1 Introduction.................................................................................. 11
  4.2 Requirement Analysis.................................................................... 11
    4.2.1 Functional Requirements of the Portal...................................... 11
    4.2.2 Non-functional requirements of the Portal................................. 14
    4.2.3 Feasibility Study................................................................. 15
  4.3 Architectural design of the Portal.................................................... 16
  4.4 Object-Oriented approach to Portal components design................ 18
    4.4.1 Analysis level class diagram................................................ 18
    4.4.2 Use Case Realisation......................................................... 19
    4.4.3 Design level class diagram................................................ 19
  4.5 Design of the Portal Interface....................................................... 20
  4.6 Summary..................................................................................... 21

Chapter 5. Implementation of the Portal.................................................. 22
  5.1 Introduction.................................................................................. 22
  5.2 Existing Approaches to Implementation of Web Portals................... 22
    5.2.1 JavaCoG Kit & GPDK............................................................ 22
    5.2.2 JetSpeed, WebSphere & GridSphere........................................ 22
    5.2.3 Struts & GridScape............................................................. 23
    5.2.4 Perl Portal............................................................................ 23
  5.3 The JSP/JavaBean Solution............................................................. 23
  5.4 Implementation............................................................................. 24
    5.4.1 JavaBeans........................................................................... 24
    5.4.2 JSP pages........................................................................... 26
  5.5 Testing of the Portal....................................................................... 28
  5.6 Summary..................................................................................... 28

Chapter 6. Evaluation............................................................................. 29
  6.1 Introduction.................................................................................. 29
  6.2 Portal Evaluation........................................................................... 29
    6.2.1 Criteria................................................................................ 29
    6.2.2 Functionality........................................................................ 29
    6.2.3 Performance......................................................................... 30
    6.2.4 Interface............................................................................. 30
  6.3 Project Evaluation.......................................................................... 31
    6.3.1 Criteria................................................................................ 31
    6.3.2 Objectives.......................................................................... 31
    6.3.3 Learning............................................................................. 31
    6.3.4 Management...................................................................... 32
  6.4 Minimum Requirements.................................................................. 33
  6.5 Summary..................................................................................... 33

Chapter 7. Conclusion............................................................................ 34
  7.1 Introduction.................................................................................. 34
  7.2 Project Achievements..................................................................... 34
  7.3 Possible Enhancements and Improvements..................................... 34
  7.4 Project Summary.......................................................................... 34

Appendix A. Personal Reflection.............................................................. 35
Appendix B. MSc Project Marking Scheme, Interim Report Comments, Project Objectives & Deliverables.................................................. 36
Appendix C. Use Case Realisation............................................................ 39
Appendix D. Design of the User Interface ................................................................. 42
Appendix E. Test Cases .......................................................................................... 45
Appendix F. Presentation Slides ............................................................................ 50
References .............................................................................................................. 58
Table of Figures

Figure 2.1 System overview of Globus Toolkit ................................................................. 5
Figure 3.1 Nimrod/G Components Interactions ............................................................. 8
Figure 4.1 Use Case Diagram for Nimrod/G Portal ....................................................... 12
Figure 4.2 Create New Experiment Use Case ................................................................. 12
Figure 4.3 Monitor Experiment Use Case .................................................................... 13
Figure 4.4 Nimrod/G Portal Architecture ...................................................................... 17
Figure 4.5 Nimrod/G Portal Design ............................................................................. 18
Figure 4.6 Analysis level Class Diagram ....................................................................... 18
Figure 4.7 Design-Level Class Diagram ....................................................................... 20
Figure 4.8 User Interface for Create New Experiment .................................................. 21
Figure 5.1 Implementation Architecture of the Portal .................................................. 24
Figure 5.2 Hierarchy View of Portal Pages .................................................................... 26
Figure 6.1 Approved Project Schedule .......................................................................... 32
Figure 6.2 Actual Project Scheduling ........................................................................... 33
Chapter 1. Introduction

1.1 Project Aim

The last decade of years has seen the dramatically rapid development of the Internet along with the growth of its related technologies. The Grid, rested on the Internet, has promised to provide infrastructure services for a new generation of computing. Today the Grid sees its application in areas of distributed computing, large-scale data analysis, parametric study, collaborative working and so forth.

Key to the power of the Grid computing is the management and scheduling of computational resources. To help facilitate this, a variety of Grid Resource Brokers and schedulers have been developed in the industry. Nimrod/G, with its declarative parametric modelling language and an economy-based, flexible scheduling scheme, is regarded as the super scheduler and important tool for the research of parametric study and computational economy.

Although Nimrod/G has provided a client GUI tool for experiment control and monitoring, a Web based approach to the creation and monitoring of parametric experiments is not available. The aim of the project therefore is to implement a Grid portal as a convenient interface that masks the complexity of Nimrod's parametric modelling language. Web-based access to the Grid has many potential benefits to the user: the portal is accessible at all time worldwide; the user need not install any software but the “thin client” browser; a portal interface is more intuitive for the user, etc. It is also anticipated that a set of Grid technologies be learned from this project through the design and implementation of the portal.

1.2 Project Objectives

1.2.1 Understand Grid Computing

In order to be able to design a Grid portal it will be necessary to gain a full understanding of how a Grid works. There is a lot of active research on Grid. We shall look at some of the important factors in this field:

- Grid definition
- Major issues for Grid computing
- How Grid tackles these issues
- Recent developments of the Grid

1.2.2 Understand Grid Resource Management and Scheduling

To be able to master resources from a portal it will be necessary to know details of resources management and scheduling. We shall have in-depth knowledge of:

- Why resource management and scheduling
- Protocol for managing and scheduling resources on the Grid
- Existing Grid Resource Brokers and how they work

1.2.3 Study Nimrod/G

A full study of Nimrod/G is necessary before we design a portal for it. For Nimrod/G we need to know:
- Its parametric language
- The Nimrod implementation of computational economy
- Its APIs

1.2.4 Design a portal for Nimrod/G

Before the implementation of the portal we need a good design to guide us through. The design solution should provide:

- Requirements of the portal
- Architecture of the portal
- User Interfaces designs

1.2.5 Implement the portal on grid test bed

We should think about different ways for portal implementation and implement the designed portal with one approach on a test bed. The implemented portal should also be tested.

1.2.6 Integrate the Portal with Nimrod/G

The implemented portal should finally be linked to Nimrod/G meet the aim of the project.
Chapter 2. Grid Computing and Globus Toolkit

2.1 Introduction

The background research of this project will present a structured introduction to the world of Grid Computing, of its definition, key concepts, important areas and recent developments. One of its key areas, Grid resource management, will be examined in the next Chapter, as this reflects the thesis of this project.

2.2 The Grid

The Grid is getting more and more popular a concept in the industry. Optimists are welcoming the Grid as the next generation of Internet. The infrastructure nature of the Grid was first identified in 1998 as:

“A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.” [1]

This concept had later evolved from the four fundamental characteristics to be a definition of “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations” [2]. The focus of the latter definition has been the negotiation among the participating parties, which led to the more recent and service-oriented definition of:

“A Grid is a system that coordinates resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service.” [3]

To better understand the Grid concept we examine some important Grid notions:

1) **Resources**: can be computation power, communication resources, instruments, storage or data for sharing. Also often referred to as “nodes”, “members”, “clients”, “hosts” or “engines”.

2) **Service**: process or system providing a facility to the network [RFC 2165].

3) **Virtual Organization**: “dynamic collections of individuals, institutions and resources” [2] that share resources on the Grid.

4) **Grid problem**: “flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions and resources” [2].

5) **Grid computing**: the process of solution finding for Grid problems.

6) **Grid technologies**: protocols, services and tools that facilitate Grid computing. Challenging technologies, including security solutions, resource management protocols, information query protocols and services and data management services are explained in [2].

There are four important issues in Grid computing that every Grid problem is concerned.

2.2.1 Security

A robust security infrastructure is “key to the success or failure of a Grid environment” [4]. The complex nature of the Grid requires “autonomously administered” [4] diverse resources to be shared in a protected manner that “different security requirements” [5] and “possible conflicting
security policies” [5] from different organizations can be satisfied. The Grid is most imperatively concerned with two security issues:

1) Authentication: the process of “verifying the identity of a participant to an operation or request” [5]. Grid users need to be authenticated when requesting a Grid service and mutual authentication of the servers is equally important.

2) Authorization: based on authentication, this process checks the access control list to determine whether a particular operation is allowed for the authenticated user.

Other security issues for the Grid may involve accounting, audit, integrity and confidentiality [5].

2.2.2 Resource Management and Scheduling

Resources in a Grid is highly distributed and diverse in form therefore “integrating, coordinating and presenting them as a single resource to the user” [6] is clearly important. Different Grid applications will share resources simultaneously and each application will “seek to leverage the performance potential of the Grid to optimise its own execution” [7] therefore scheduling is “fundamental to performance” [7]. We shall cover this aspect of the Grid in detail in Chapter 3.

2.2.3 Information Service

For a dynamic environment like Grid, information is changing constantly. Information Service provides a mechanism for registering and requesting up-to-date information about “the structure, resources, services, status and nature of the environment” [4]. Directories need to be set up to provide and maintain the above-mentioned information.

2.2.4 Data Management

Access to resources on the Grid can involve the access to giga-bytes or even tera-bytes of data. This demands the Grid to be capable of regulating and optimising various storage systems to achieve satisfying performance for data access. We also need Meta-data Services to provide information about the storage systems so that a data operation knows where and how to access to the requested data. For data-intensive science applications on the Grid, these requirements are essentially important.

2.3 Globus Toolkit

The Globus Toolkit [8] is the de facto standard for deploying Grid systems worldwide. For the Grid to support a wide variety of applications and programming paradigms, the Globus Toolkit provides a “bag of technologies” [8] rather than a unified programming model. These technologies are provided as a set of useful components that can be used independently or together to implement the Grid services such as security, resource management, information services and data management, as illustrated in Figure 2.1.
2.3.1 Grid Security Infrastructure (GSI)

GSI is an infrastructure that underlies all other component sets to provide security functions including single/mutual authentication, confidential communication, authorization, and delegation. It is based on existing protocols of SSL protocol (Secure Socket Layer), public key encryption, and x.509 certificates to provide “a single-sign-on, run-anywhere authentication service, with support for local control over access rights and mapping from global to local user identities” [8].

2.3.2 Grid Resource Allocation Manager (GRAM)

This protocol provides resource allocation and process creation, monitoring, and management services. It maps RSL (Resource Specification Language) requests into commands to local job scheduling systems to request and use remote system resources [8]. It also “provides a simple authorization mechanism based on GSI identities and a mechanism to map GSI identities to local user accounts” [8]. Note that GRAM does not provide scheduling or resource brokering capabilities and there exists a great number of high-level Resource Brokers and Schedulers that implement GRAM, which we shall examine in Chapter 3.

The IBM Redbook [9] describes how GRAM works:

“When a job is submitted by a client, the request is sent to the remote host and handled by the gatekeeper daemon located in the remote host. Then the gatekeeper creates a job manager to start and monitor the job. When the job is finished, the job manager sends the status information back to the client and terminates.”
2.3.3 Monitoring and Discovery Service (MDS)

MDS provides access to “static and dynamic information of resources” [9]. Also known as Grid Information Services (GIS), this component framework provides “information about Grid resources for use in resource discovery, selection, and optimization” [8]. MDS has the following components [9]:

1) **Information Provider**: gather resource information and pass it to GRIS.
2) **Grid Resource Information Service (GRIS)**: the repository of local resource information derived from information providers. Can register its local information with the GIIS.
3) **Grid Index Information Service (GIIS)**: the repository that contains indexes of resource information registered by the GRIS and other GIISs.
4) **MDS client**: gets the resource information directly from GRIS (for local resources) and/or a GIIS (for grid-wide resources).

To enable access control to information, integration with the GSI is made possible.

2.3.4 Grid File Transfer Protocol (GFTP)

Secure and reliable data transfer among grid nodes are enabled by the GFTP protocol, which is based on FTP and optimized for high-bandwidth WAN. The GFTP tools are also used by GRAM to transfer the output files from servers to clients [9].

2.4 Future development of the Grid

The Grid world has seen and is seeing more and more service-oriented development in terms of technologies. The protocol-oriented approaches to solve Grid problems have been changed to the higher-level service oriented ones. The industry is also researching into and is succeeding from the commercial application of the Grid [10]. However, since the project will work closely with the existing Nimrod/G software, which is a GT2 application, details of these trends are beyond the scope of this project.

2.5 Summary

We have presented an overview introduction to the Grid and a basic study of the Globus Toolkit and it’s implementation of the Grid issues. We conclude the chapter with a brief discussion of the recent development of the Grid.
Chapter 3. Grid Resource Management & Scheduling

3.1 Introduction

As noted in section 2.4.2 the GRAM protocol does not management Grid resources by itself. Instead, Grid resource brokers (GRB) are designed to perform this task. We first study what GRBs are and see some examples, from where we come to introduce the broker software of Nimrod/G and the notion of Grid Portal.

3.2 Grid Resource Brokers

Working as a mediator between the user and the Grid resources, GRB is responsible for 1) resource discovery, 2) resource selection and trading, 3) the binding of application, data, and hardware resources, 4) the initiation of computations and any required adaptation to changes in the Grid resources and 5) collection of results back to the user [11]. When the broker receives a high-level request from a Grid user, it translates the request into low-level Globus calls and utilise the Grid Information Services to query information from the resource infrastructures and finally feedback the information to the user.

Grid resources are shared by different Grid applications. In order to avoid overload to a certain resource, scheduling the resources is essentially important. GRB uses schedulers for Grid resource scheduling. Resource schedulers will try to coordinate multiple requests for access to a given resource “by optimising fairness criteria or resource utilization” [7]. Other-purposed schedulers include job schedulers, which aim to promote the system performance by optimising job throughputs, and application schedulers, which promote the performance of individual applications using “application centric cost measures” [7]. The goals of these schedulers are usually conflicting and therefore need careful balancing.

There are numerous GRBs existing in the literature, we give introduction to some and examine the Nimrod/G GRB in the following section.

3.2.1 Sun Grid Engine (SGE)

SGE [12] provides an engine for the users to submit jobs to the Grid. When a new job is submitted to the engine, a requirements profile for that job will be created first. SGE then monitors the resources to see if there are available nodes. When a queue is ready for a new job, suitable jobs for that queue will be determined and dispatched for execution. Least loaded queues will always be taken for new jobs first.

3.2.2 Condor-G

Condor-G [13] provides a “powerful, full-featured task broker” for managing both inter-domain resources using the Globus Toolkit and intra-domain ones using Condor [14]. Like SGE, it uses a queue to handle job submission. It starts a job on the remote machine using Globus Toolkit and provides interface to manage the jobs through Globus control commands or Web portals.

3.2.3 Portable Batch System (PBS)

PBS [15] is a queuing system originally developed for NASA. The purpose of PBS is “to provide additional controls over initiating or scheduling execution of batch jobs” and “to allow routing of those jobs between different hosts”. One reason for choosing PBS lies in its flexibility in allowing
different sites to establish their own scheduling policies and use different scheduling algorithms for running jobs.

3.2.4 Load Sharing Facility (LSF)

LSF [16] is a commercial GRB from Platform Computing Corp. that comprises distributed load sharing and batch queuing functionalities for managing, monitoring and analysing the resources and workloads on a network of heterogeneous systems. Its “self-healing and self-adaptive capabilities” provide fault tolerance capabilities and ensures the “always-on availability of resources”.

3.3 Nimrod/G

Nimrod/G [17] is a GRB for creating parametric experiments with novel resource management and scheduling algorithms based on economic principles. It consists of four components [18]:

1) A Task Farming Engine (TFE) that interacts with the scheduler and dispatcher to manage and control an experiment.
2) A scheduler that performs resource discovery, trading and scheduling.
3) A dispatcher and actuators that deploy agents on Grid resources to execute resource-mapped jobs. The dispatcher is middleware specific.
4) Agents that set up the execution environment on given resources for jobs. An agent is also responsible for transporting codes and data (input or output files) between the resource node and TFE.

![Figure 3.1 Nimrod/G Components Interactions [18]](image)

Figure 3.1 illustrates the interaction between Nimrod/G components.

Besides its layered component architecture, Nimrod/G is “novel” in many interesting aspects, as we are going to discuss in the following sub-sections.
3.3.1 Parametric Language

Nimrod/G is specifically designed to enable parameter sweep applications. It accepts the input of the plan file, which is written in parametric language, to generate jobs for an experiment. The parametric language is indeed higher-level batch utilization of the RSL language and the consequent plan file is composed of two sections, namely a) parameters section that specifies parameters and their default values and b) tasks section that describes commands for job execution.

3.3.2 Computational Economy

The scheduling algorithms for Nimrod/G are based on economic principles. User can define deadline and budget constraints for the experiment and the broker will try to find the best resources available in the Grid and use them for execution of jobs. Nimrod/G supports four scheduling algorithms, which being cost optimisation, time optimisation, cost-time optimisation and conservative time strategy.

3.3.3 Nimrod/G Clients

Nimrod/G is bundled with GUI client application tools to ease the creation, control and monitor of experiments. Before the creation of an experiment, a plan file and a gatekeeper file, which specifies Grid resources to be used, should be provided. The tool of nimrodgen can be used to generate from the plan file a run file, in which each job is mapped to a certain resource. The tool of nimrodcreate is used to create an experiment from the run file and finally the tool of nimrod can be used to start the GUI for experiment management. Running multiple instances of the same client at different locations is possible.

3.3.4 Nimrod/G APIs

The TFE of Nimrod/G provides a set of APIs for external programmes to interface with the broker. The interface is handled via a TCP/IP protocol. At the time an experiment is created (i.e. the command of nimrod is executed), the Dispatcher prints out a port number of the directing interface in the experiment’s log file. The external program, which is called Director in the Nimrod world, can then connect to the port to establish a TCP communication with the Dispatcher, send Directing API commands to it to proceed with querying about execution progress or control the execution [19].

3.4 Grid Portals

A Grid portal is “a Web-based application server enhanced with the necessary software to communicate to Grid services and resources” [22]. Doing a Grid portal is a process of mapping and visualizing the GT or GRB functionalities onto comprehensive and intuitive Web pages. There are several reasons for developing Web-based portals for the Grid:

1) The Globus Toolkit and GRBs that implements it provide powerful protocols and services to facilitate Grid computing but do not have easy and simple interface to the user.
2) Low bandwidth and high latency WAN requires the client brokers to be lightweight ones. Web-based portals are lightweight implementations.
3) End users of the Grid need to see customisable, simple and friendly interfaces.
4) The Web Service nature of OGSA enables Grid functionalities to be best incorporated a Web portal with relative ease.
5) Some GRB applications such as Nimrod/G client can be implemented into Web portals to provide browser-based view of Web access and avoid installation of the client software.
Grid portals can be subdivided into two categories, namely application-specific (e.g. ASC Portal [20]) and user-specific (e.g. HotPage user portal [21]) portals [22]. We next study some of the Grid Portals to see what Grid services are provided on the Portal.

3.4.1 HotPage User Portal

The NPACI HotPage [21] gathers information of distributed resources across member organisations to be viewed “either as an integrated Grid system or as individual machines” [11]. It provides services of 1) information services that help improve the portal usability and 2) resource access and management services that support job submission and give resource status information of CPU load, queue usage summaries, processor node maps and current queue information.

3.4.2 White Rose Grid (WRG) Portal

WRG Portal [23] is a GT2 and SGE enabled portal that provides Web accessed Globus functionalities such as user authentication, file transfer, job submission and resource query to the WRG Grid, which facilitates Grid resources from Universities of Leeds, Shield and York to do e-Science research and high performance computing. Heterogeneity of resource nodes and complexity of Globus calls are hidden behind the Portal interface.

3.4.3 Nimrod Portal

Nimrod Portal [24] is the Nimrod team’s newly developed Grid portal for Nimrod/G and Nimrod/O [17]. It can provide functionalities of portal login, experiment creation, experiment monitoring and resource management for Nimrod/G GRB. Comparison to the portal we are working on is available in the Evaluation chapter.

3.5 Summary

Grid Resource Brokers are key to the Grid resource management and scheduling. We have presented the importance of GRB followed by the study of some GRB instances. Details of Nimrod/G are provided as background knowledge for our system design and implementation and the need for Grid Portal has also been covered. Next we begin to design and implement a portal for Nimrod/G.
Chapter 4. Design of the Portal

4.1 Introduction

After a review of the background literatures we now come to design a solution for the Nimrod/G Portal (hereinafter referred to as Portal). We shall first analyse the problem we are facing, and then provide a design solution for the Portal, including the design of Portal architecture, components and interface.

4.2 Requirement Analysis

One of the best ways to design a problem-solving system is to first study the system requirements, i.e. what we expect the system to do for us. The ultimate aim of designing a web portal for Nimrod/G lies in the complexity and inconvenience when running Nimrod application for experiment creation and monitoring. As discussed in Section 3.3, creating experiments and monitoring experiment jobs with Nimrod/G application not only require the users, usually scientists, to first log on into a Nimrod enabled Grid test bed but demand the users to be at least aware of the commands and procedures for experiment generation and application execution. The Portal’s job is to mask these complexities and provide a simple web interface for Nimrod/G users to create and monitor their Nimrod experiments via a Web Browser with relative ease. Similar to the Nimrod/G application, should functionally include:

1) Pages for users to create experiments
2) Pages for experiment monitoring
3) Pages for setting QoS parameters for better scheduling or rescheduling
4) Pages for the administrator to set up resources/nodes for experiments.

There are also non-functional aspects to be considered, which are equally important for Portal design. We first begin with Use Case analysis.

4.2.1 Functional Requirements of the Portal

As shown in Figure 4.1 there are two categories of Portal users: ordinary users and Grid administrator. The latter is responsible for managing Grid resources/nodes for experiments and managing user account information. The Portal needs to interface with Nimrod/G Dispatcher, which provides a unique contact port number for each Nimrod experiment and dispatches experiment jobs to the Grid nodes for execution facilitating Nimrod’s parametric logic.

Details of each Use Case are discussed in the following three sections.
4.2.1.1 Create New Experiment

The creation of a new experiment (Figure 4.2) mainly consists of three steps: firstly the user needs to create a plan file for it; secondly he/she may need to upload experiment files to the Nimrod/G server and finally he/she specifies resources/nodes to be used for that experiment. The first (generation of a plan file) and third (generation of a gatekeeper file) requirements are compulsory while the second one is optional for the creation of an experiment.
4.2.1.2 Monitor Experiment

Monitoring an experiment is indeed a general concept. When we say a user is monitoring an experiment on the Grid, he/she has actually interest in either one or more aspects of the experiment, which for a Nimrod/G experiment should include (Figure 4.3): -

- **Experiment status:** is the experiment started/inactive/done? In addition the user may be interested to see how long the experiment has been executing or how much time it takes to have an experiment done.
- **Plan file:** displaying this to the user will remind him/her what jobs the experiment is currently doing or has done. The user will need to modify the plan file as well if applicable prior to the experiment execution.
- **Experiment files:** the user may have specified file operations (e.g. copy) in the plan file and uploaded necessary files to be processed. Therefore it makes sense for the user to have access to a list of experiment files including user uploaded ones and experiment output files. It is also necessary to provide a channel in which the user can download the output files to his/her local machine.
- **Job information:** this refers to the job instead of experiment status, including information on how many jobs are currently running and how many/which jobs are ready/executing/done/failed. The user would also like to see details of each job such as job parameters and job server information if applicable. He/she would also like to be provided with a way to reset job status, e.g. set a failed job status to ready so it can try running for a second time.
- **Schedule type:** this will provide the user of the schedule type specified when creating a new experiment.
- **Resources used:** a list of resources being used for this experiment.

4.2.1.3 Other Use Cases

- **Control Experiment:** User of the Portal will need a mechanism to start/pause/resume/archive the experiment.
- **Set QoS Parameters**: This is important to Nimrod/G as it reflects the GRB’s *computational economy* concept. The Portal should allow users to set deadlines and budgets/costs for experiments and make use of the Nimrod algorithm for economic scheduling.

- **Resource Management**: the Administrator should be able to manage resources (i.e. nodes) for experiment execution. This could be automatic resource discovery with MDS or manual addition/deletion of Grid nodes in a default gatekeeper file for setting up experiment resources.

4.2.2 Non-functional requirements of the Portal

Major concerns of the design of the Portal regarding Non-functional requirements include Security, Performance and Usability.

4.2.2.1 Security

1) **Login to/Logout from the Portal**

The user should be authenticated before accessing Grid resources for experiments. During a login session, the same authenticated user does not have to be re-authenticated again while navigating from one page to another. Likewise, when the user logs out of the Portal the session should be terminated.

2) **User Account**

A Database is necessary to hold user specific information and separate one user’s experiments from each other. Giving user specific information can also avoid showing all experiment contents to the user, hence will enhance performance as well.

It should be noted that an qualified user accessing the Grid resource should be authenticated not only in the Web Portal database but should also be an authenticated user of the Nimrod/G server, i.e. he/she should have the necessary authentication to log in to the Nimrod/G server machine. The reason for this lies in the Grid’s high demand on security.

3) **Concurrency Control**

When more than one user is accessing the Portal to create new experiments, concurrency may be a problem. Synchronized access should be made possible facilitating a thread-enabled implementation plan.

4.2.2.2 Performance

With the prototype Portal this is not a major concern but there are still points to note. The Portal should not be too slow to be acceptable when displaying experiment information to the user. The Portal should also be capable of serving a relatively large number of online users at the same time. However this will largely depend on Nimrod/G’s scalability and stability, which is beyond the main scope of this project.
4.2.2.3 Usability

This is concerned with Human-Computer Interaction. The interface of the Portal should be ‘usable’, which requires all Portal functions be available from the interface. In order to make it fast and easy to accomplish jobs on the Portal, it is expected to have the Portal interface designed in a direct, concise and integrative manner.

After revising lecture notes [25] from the ISE module, I take Learnability, Flexibility and Robustness as the main principles when considering usability of the Portal. Learnability requires the Portal to be consistent, i.e. having some likeness from pages. Flexibility demands the system to have certain degree of freedom for navigation and at the same time avoiding undesired errors. Robustness requires the Portal to be responsive to the user’s operations.

4.2.3 Feasibility Study

After studying the system requirements we now decide which requirements are applicable. First I shall name some of the constraints that we are facing during this stage of the project.

4.2.3.1 Time Constraint

At the time I finished the background literature review there are only 7 weeks left for designing and implementing the prototype Portal. It would not be possible to design and implement the Portal to meet all the requirements within the rest of time. After a discussion with my supervisor, I was finally convinced that I would have to make the Portal design as simple as possible and implement as much functionality as I could.

4.2.3.2 Technique Constraint

The whole year of my MSc study has given me familiarity with Java programming, Network Programming, Database Design & Programming and Information System Design, but I do not have any experience working with Portal design and implementation. Whichever way I choose to implement the Portal, I will have a lot of techniques to learn. This makes it not possible for me to design the Portal to employ professional skills in those adopted techniques, and using simple techniques and basic skills will not realize many complex functionalities for the Portal.

Giving up those over-demanding functionalities would be a good decision. Since I have really no knowledge of how the Linux world provides secure accesses to the system, the first requirement I decided to postpone for future development is the secure user account management module.

4.2.3.3 Nimrod/G Constraints

Working with Nimrod/G is not an easy job. Most of my design and implementation problems come with the constraints from Nimrod/G. There are three key constraints from Nimrod:

A. There is NO Nimrod/G API documentation available!

The Nimrod/G software package only contains a guidance file explaining how to install Nimrod/G, how to write up the plan file and how to run an experiment with the bundled Nimrod/G application. It does not contain any documentation or introduction to the Nimrod/G APIs. The same happens on the Nimrod/G website [17]. Hopefully there is a User Guide [26] online for Nimrod’s commercial counterpart enFuzion, from which I found Chapter 9 explaining how to call enFuzion’s API from external programmes using Director commands. It was later when I got into contact with Professor David Abramson, who is in charge of the Nimrod/G project team in Monash University and kindly had my email requests answered by
his research scientist team members Dr. Colin Enticott and Mr. Slavisa Garic that I begin to realize how I could possibly design and finally implement the Portal.

B. Nimrod/G dispatcher only accepts local connections.

Following my contacts with the Nimrod team, I learned to write a Director programme to talk with the Nimrod Dispatcher, receive responses from Dispatcher. The Nimrod team’s answer to my question “why Nimrod/G Dispatcher does not accept remote connections” is that it only accepts local connections due to the research instead of commercial nature of Nimrod/G. This means the Nimrod Server, which contains the Nimrod/G Dispatcher that has full access to Grid nodes, will have to be the same machine as the Web Server, which contains the Portal. This has indeed constrained the Portal to be both inflexible and insecure for implementation.

C. Nimrod does not provide API for QoS control.

After a brief study of the G-Monitor source codes [27] (in Perl) and the API commands that Nimrod or enFuzion provides, I found that G-Monitor Portal does not use any Nimrod API to get experiment deadline or budget information, instead it uses a complex algorithm in Perl to realize the QoS functions and do the re-scheduling of experiment jobs for Nimrod/G. I do not want to provide an algorithm of my own to cope with QoS parameters, nor do I want to interpret the Perl code and convert it into the implementation language I am going to use. Given the time and technique constraints discussed in the above sections, it would be a good decision to exclude the “Set QoS Parameters” module from the Portal design.

Based on the discussion of constraints above, feasible Portal modules that SHOULD be included for this project can be identified as the following list:

- The full functioning Create New Experiment module with a concern of possible concurrency issue when creating new experiments.
- Monitor Experiment module: excluding the Schedule Type function.
- Control Experiment module: providing control of experiment status.
- Resource Management module: the Portal should at least provide manual addition/deletion of nodes for resource management.

It is also decided that the project WILL NOT involve the implementation of:

- The Set QoS Parameters module: as this is not yet mature for implementation.
- User Login/Logout modules: as these will involve user authentication at system level.

4.3 Architectural design of the Portal

The Portal sits between the client browser and Nimrod/G server to provide web access to Grid resources. It interacts with the Nimrod/G server through the implementation of Director-Dispatcher mechanism. A Director sends commands and requests to the Dispatcher in the Nimrod-G server to make use of the Nimrod-G APIs. The Dispatcher, upon receipt of the Director commands and requests, dispatches experiment tasks to or queries information from the Nimrod-G TFE. The TFE will then assign jobs to or gather information from Grid resources (node machines) through the Globus GRAM protocol, feedback to the dispatcher and the latter will return results to the Director of the Portal.
As visualized in Figure 4.4, this design scheme shows the Portal to be tightly coupled with Nimrod APIs, i.e. the Portal will function as long as Nimrod API supports the relevant functionality. Therefore, the core of the Portal design is the communication between the Portal side Director and the Nimrod/G server side Dispatcher. The TCP/IP connection between them indicates that socket programming is required for the implementation of the Portal, which is again totally new to my knowledge.

A Director-Dispatcher coupled solution can facilitate the Nimrod APIs to query on or control experiment status and job status. However, it cannot handle the creation or removal of an experiment (as either task is performed outside the Nimrod GRB), nor can it be used for resource management (as the gatekeeper file is created prior to the creation of an experiment). This indicates that a new module should be introduced into the Portal design to “talk” to the Nimrod Server system Shell to create/remove an experiment or manage the gatekeeper file. The ShellAgent module as shown in Figure 4.5 will do this job. It should be noted that such a module should be capable for running system processes and invoking Linux Shell commands to create/remove experiments or generate the gatekeeper file.
4.4 **Object-Oriented approach to Portal components design**

Based on Figure 4.5 design scheme, we can finally start to analyse the internal components of the Portal system. From an Object-Oriented approach, we first need to identify system classes at a concept level, and secondly carry on with Use Case Realisation to further sort out class attributes and methods as well as the relations among classes; finally we shall have a detailed design-level class diagram ready for implementation.

4.4.1 **Analysis level class diagram**

Some conceptual system classes can already be identified. As shown in Figure 4.6, an experiment will involve the scheduling of a group of jobs, the input of the initiative plan file, the uploading of experiment files and downloading of the output files, and finally the selection of a list of Grid resources. To access Nimrod/G using a web-based Portal, the Experiment module first needs to talk with the Director module to retrieve job/experiment status information, then needs the help of ShellAgent module to generate new experiments and manage resources. To gain an idea of how these classes will interact with each other we need to do a Use Case Realisation.
4.4.2 Use Case Realisation

The aim of this section is to use the Use Case Realisation technique I learned from the ISE module to further identify attributes and methods within each class and relations among them. New classes will be introduced when necessary and unwanted ones will be removed from the class diagram.

Full process of the Use Case Realisation can be accessed in Appendix C using Collaboration Diagrams. It should be noted that the process is an iterative one that involves two iterations, and the results shown in the appendix is the ultimate outcome of the Realisation process, i.e. the outcome of the second iteration.

The iterations has achieved satisfying results, as listed below:

1) Split the *Experiment* class into a) *Experiment*, which will inherit from the *ShellAgent* class for new experiment creation and resource management and b) *ExperimentDirector*, which will specifically function as the Director to communicate with the Dispatcher.
2) Introduce a *JobDirector* class that will inherit from *Director* to handle job information with the invocation of Nimrod APIs.
3) Combine the *PlanFile* and *Files* classes into one *ExperimentFile* class, which emphasizes from the Object-Oriented perspective that both the plan file and experiment files are by nature in common and therefore we can use similar methods to operate on them.
4) Class attributes and methods become clear and are conceptually proved.

From these outcomes it finally make it possible to generate a Design level Class Diagram for the Portal system with relative ease.

4.4.3 Design level class diagram

The diagram is shown in Figure 4.7. This diagram is in such a detailed and conceptually proved manner that it has actually made the implementation of the Portal on its way.
4.5 Design of the Portal Interface

Before implementing the prototype Portal, we should have the UI prototypes ready first. Since the Portal is purely web-based, we can use HTML pages to show the prototype interface. As discussed at the beginning of section 4.2, we need at least design three pages for the modules of 1) Create experiment 2) Monitor experiment 3) Manage resources. For ease of use we split the interface into two frame columns (Figure 8) and put a navigation page on the left hand side providing links for quick switching between different functional module pages. The right hand side is designed to be the main frame that will display information upon user requests.

Figure 4.8 shows an Interface for Create New Experiment page. The interface for Monitor Experiment page is relatively complex. It needs to be a comprehensive page integrating the five sections of experiment status, plan file, experiment files, job information and resources used. The Experiment Status section will also have links for Experiment Control. Moreover, user can click on a file listed in the Experiment Files section to download it and can click on a job from the Job Information section for Job Details and Job Status Control. The Manage Resource page, on the other hand, simply provides 1) a combo box for selection of node type and a text field for input of new resource node address to add node and 2) a list of existing nodes for the administrator’s choice to delete node. Detailed interface design for the latter two pages is available in Appendix D.
In this chapter we followed an object-oriented approach to identify system requirements and propose design solution for system architecture. Next we did Use Case Realisation to sort out system component classes and finally we designed the HTML interfaces for the Portal. The design solutions achieved at this stage are blue prints that will guide us through the implementation of the Portal.

4.6 Summary
Chapter 5. Implementation of the Portal

5.1 Introduction

We have already had a detailed design solution for the Portal, now we shall try to make it come true.

We first discuss existing approaches to the implementation of Web Portals, from where we introduce our simple JSP/JavaBean solution and then carry out the implementation with it. After the implementation we need a good testing of the prototype to see if the designed features have been achieved.

5.2 Existing Approaches to Implementation of Web Portals

5.2.1 JavaCoG Kit & GPDK

Commodity Grid Kit relates the commodity technologies to the Grid world [28]. It is defined as “sets of general components that map Grid functionality into specific commodity environments or frameworks”. Java Commodity Grid Kit [28] provides access to the Globus Toolkit functionalities through Java framework. The JavaCoG Kit not only provides mappings to the Grid Services but also has low-level utility components to extend the Globus toolkit functionality and provides GUI components for application developers [28].

GPDK [29] is a toolkit that facilitates the JavaCoG kit to provide both “a portal development environment for the creation of new portals” [29] and “a collection of Grid service beans used to accomplish basic operations such as job submission, file transfer and querying of information services” [29]. Following the Model-View-Controller (MVC) design pattern, it uses PE (Portal Engine), APOs (Action Page Objects) and VPs (View Pages) to directly make use of the Grid service beans provided by Java CoG kit [30]. It relies on “commodity Java technologies including Java beans, servlets and JSPs” [30] for implementation. The White Rose Grid Portal [31] is a good example Grid Portal that is GPDK based.

5.2.2 JetSpeed, WebSphere & GridSphere

Although GPDK can be used to create project specific Grid portals, it has been found with a major limitation of “lack of any reusability in the presentation layer” [32]. The frameworks we are going to introduce, especially GridSphere [36], aim to address this problem. Portlets [33] are the building blocks for these frameworks. With the analogous functionality to servlets, portlets are managed by the portlet container to enable the portal to aggregate contents from different sources into one portal page, allowing the page to be more customisable.

JetSpeed [34] is the first framework that specifies and uses portlets. It is designed for rapid development of XML-based, content-independent portal applications using portlets. The IBM WebSphere [35] is a similar but commercial version of framework that offers “a more coherent model for describing portlets” [32].

The GridSphere [36] framework by GridLab [37] seeks to tackle this problem and “offer external developers a model for easily adding new functionality and hence increasing community collaboration” [32]. GridLab’s vision is to integrate the GridSphere framework with OGSA to provide services that have access to Globus based Grid services such as resource brokers, monitoring and data management [32].
5.2.3 Struts & GridScape

Struts [38] is a framework that provides “invisible underpinnings” [38] for Web applications. It bases on standard technologies such as Java Servlets, XML, JavaBeans, JSPs to construct web applications with the Model 2 approach, which is an advanced version of the MVC design paradigm. The vision of the framework is to achieve easier maintenance and greater modularity of the application by completely separating Models from Views. With Struts, working on business logic and user interface in parallel is possible.

GridScape [39] is a Grid-aware tool that facilitates Struts for fast implementation of Grid portals.

5.2.4 Perl Portal

Besides the Java-based frameworks, there are ad-hoc approaches to the realisation of a Web portal. G-Monitor [40] is a Perl implementation of the Nimrod/G Portal. The solution provided in G-Monitor is both project and application (Nimrod/G) specific.

5.3 The JSP/JavaBean Solution

The framework approaches discussed above are too intricate and impractical to apply to the Nimrod/G Portal, which is based on the interaction between Directors and Nimrod APIs, other than low-level Grid services. The Perl approach is not suitable for me as I am not aware of the language. Although Perl is simple for mixing HTML texts with CGI codes, it cannot effectively tell the Models apart from the Views hence is hardly extensible and not suitable for future maintenance.

Given the Portal’s simple structure and the Nimrod API-based solution provided in the Design chapter, it would be a better idea to apply a simplified Struts solution that is Object-oriented in nature and can benefit from the MVC model and at the same time avoids the complexity for deployment. The JSP/JavaBean solution best suits into this context.

Learning and programming with JSP [41] and JavaBean [42] are also more motivating for me, given my background and interests in Java. Details of the JSP and JavaBean technologies are beyond the scope of this project though.

This solution requires a Tomcat [43] server to be set up and work as the JSP (servlet) container. Due to the Nimrod/G constraints, it has to be deployed in the Nimrod/G server machine to handle local host only connections. Figure 5.1 shows the architecture of the Portal for implementation. Note that the Nimrod/G Server means the machine that has Nimrod/G installed and in our case, it has actually become part of the Tomcat Server machine.
5.4 Implementation

The implementation of the Portal involves two stages. One is the implementation of the JavaBean Models and the other is the implementation of the JSP Views.

5.4.1 JavaBeans

We follow our blueprint Class Diagram (Figure 4.7) to implement the system beans.

5.4.1.1 TelnetAgent class & Director beans

The TelnetAgent bean makes use of the JScape iNet Factory [44] Telnet API to 1) make TCP connection to a remote server port and 2) send message strings to and receive responses from the server. To make a connection, it simply use the package to create a new Telnet instance and establish the connection, which goes:

```java
Telnet telnet = new Telnet (hostName, portNumber);
telnet.connect();
```

The method `getResponse()` is used to perform the second function.

The Director bean facilitates the TelnetAgent to make connection to a Nimrod/G Dispatcher port, which is unique for each experiment, and send command strings to and receive results from the Dispatcher. To send a command to the Dispatcher and get back results, we can simply use:

```java
setHostname(hostAddress);
setPort(portNumber);
setCommand(commandString);
String result = getResult();
```

The ExperimentDirector and JobDirector beans are all extensions to the Director class. Implementation of the JavaBean methods for these two beans largely relies on the Nimrod API
commands (as listed in Section 3.3.4). Besides using the Directing Commands for experiment status query and control, the ExperimentDirector also use the “setkey” command to set experiment start time and use “getkey” to get it back and calculate experiment elapsed time. The JobDirector bean makes full use of the Nimrod/enFuzion Job Commands to query job information and perform job status control.

5.4.1.2 ShellAgent class & Experiment bean

The ShellAgent is a class for the execution of commands in runtime shell. Executing a runtime command in Java is handled by the Runtime class [45]. According to the Java API Specification [45], “Every Java application has a single instance of class Runtime that allows the application to interface with the environment in which the application is running. The current runtime can be obtained from the getRuntime method.” The execCommand(String command) method is used as a core function in ShellAgent to delivery the command string from the Portal to the Shell environment. Learned from an online article [46], I put only one line in the method’s body:

```
Process cmd_proc = Runtime.getRuntime().exec(command);
```

We may need to capture the outputs of some command executions, for example when we execute the “/bin/ls” command in an experiment directory the Shell will output a list of files in the current directory, which we can capture and display to the Portal user as the list of experiment files. The getResult() method is used for this purpose. It loads the input stream of the command execution process and returns it as a String.

The Experiment bean extends the ShellAgent to realize the creation/deletion of an experiment. My algorithm for the creation of a new experiment (method newExp()) follows the list of:

1) Set an experiment name for the new experiment;
2) Make a new directory with the given name for the experiment;
3) Create a plan file with the new experiment name plus the extension of “.pln”, insert each input plan file text line into the new plan file;
4) Create a gatekeeper file for the experiment, insert each node address into the file;
5) Change to the new experiment directory; use “nimrodgen” to generate a run file, run “nimrod experimentName.run” to fire the experiment.

Together with the newly built experiment, a nimrod.log file will be created, from which we can use getPort() method to easily retrieve the Dispatcher contact port number for that experiment.

The removal of an existing experiment is no easier job. It first creates an ExperimentDirector instance to request the Dispatcher for experiment status. If the returned status indicates the experiment is not yet inactive, we need to shut it down before the operation of removal. To remove the experiment from the experiments list we use the Shell’s “/bin/rm –r –d” command to completely delete the whole experiment directory.

5.4.1.3 ExperimentFile & ExperimentResource beans

These two beans extend the Experiment class to provide file or resource specific functions. The viewFile() function in ExperimentFile executes the “/bin/cat” command to see the contents of a file (either plan file or gatekeeper file) and return the output as a String. The upload() and download() functions are actually moved into the file.jsp and download.jsp pages, as they will involve an external jspSmartUpload class from the jspSmart component APIs [47]. The ExperimentResource bean includes a series of functions to operate on the gatekeeper file, either the default one for the
String[] command = {"sh","-c","/bin/echo "+node+" >> /home/linux_b/lin/test/gatekeeper"};

Removing a node is relatively complex, the algorithm I used is:

1) Get all nodes from the gatekeeper file, store in array a;
2) Instantiate a new array b with the length equal to (a.length–1);
3) Store all nodes except the deleted one into b;
4) Remove the old gatekeeper file and create a new one with all nodes in b.

There is also a sortNode() method in the resource bean, which invokes the Shell’s sort command to operate on the gatekeeper file.

### 5.4.2 JSP pages

The implementation of the user interface is a process of converting the HTML UI design pages into dynamic, powerful JSP ones. This can be done by inserting JSP codes and tags into the HTML pages to utilize the power of JavaBeans.

Figure 5. shows the hierarchy view of the JSP pages for the Portal.

![Hierarchy view of Portal Pages](image)

### 5.4.2.1 Experiments page

This page displays a list of existing experiments to the user. It uses the Experiment bean to get the list of all experiments (getAll()) and uses the ExperimentDirector bean to fetch the status of each experiment (getStatus()) and get time elapsed (getTimeElapsed()) when necessary. The JSP page calls the beans in a form of:
5.4.2.2 New Experiment page

The newExperiment.jsp page just uses the ExperimentResource bean to show the resources list on it. Upon user’s submission it is redirected to the doNewExp.jsp page, where the user’s input experiment data are checked and, if no error occurs, handled by the Experiment bean’s newExp() method to create a new experiment. The three steps check goes:

1) Check if the experiment name is null or invalid (with system character in it) or the plan file is missing or the user has forgotten to select at least one node as resource;
2) Check if the experiment name already exists;
3) Check if an invalid plan file has led the experiment creation to a failure.

The third check is done with a expCreated() method, which returns a Boolean true if the nimrod.log file exists in the new experiment directory. If the user has selected some experiment files to upload, the doNewExp.jsp page will use the SmartUpload [47] class to handle file uploads.

5.4.2.3 Experiment Monitor page

The experimentMonitor.jsp page includes four sub-pages to give the full range of experiment information. The JSP include tag is used to incorporate the sub-pages. We can see an example of this:

```
<jsp:include page="expStatus.jsp">
  <jsp:param name="expName" value="<%=expName%>" />
  <jsp:param name="control" value="<%=control%>" />
</jsp:include>
```

The four sub-pages are:

1) expStatus.jsp: use ExperimentDirector bean to get experiment status and time elapsed and Experiment bean to get port number and control experiment. Archiving an experiment will lead to a confirm window (confirm.jsp) and if the user clicks “yes” the removeExp() method will be invoked.
2) jobs.jsp: calls the JobDirector bean to get jobs list and provides link for each job leading to a new jobDetails.jsp page, where we can see job status information and set job status.
3) file.jsp: this page is made up of two parts, one for the display of the plan file, which is handled by the ExperimentFile bean’s viewFile() function. The other is for displaying list of experiment files, which is supported by the SmartUpload class’ download function to provide link for the download of every output file.
4) expResource.jsp: requests resources being used by the experiment and queries node status with the getExpNodes() method and getNodeStatus() method respectively of the ExperimentResource bean and display them to the user.
5.4.2.4 Resource Management page

This is a simple page using the `ExperimentResource` bean’s `addNode()` and `removeNode()` functions for managing nodes in the default gatekeeper file that is used for selection of experiment specific resources for every new experiment.

5.4.2.5 Login/Logout pages

These remain HTML pages at the current stage of the project. The implementation of the login page can be done by 1) setting up a database to manage Portal user information; 2) use a session bean to store user data to allow persistence for the duration of the user's transactions on the portal; 3) destroy the user session upon user logout.

5.5 Testing of the Portal

Testing of the Portal will involve the testing of each Portal page, including the thorough testing of all elements (links, buttons, text fields, etc). Since the Portal is not a complicated website, I tested all the elements on each page and gave results of each test case in Appendix E. The 37 test cases show that most of the testing on the Portal came up with desired results, which are sometimes better than expected as shown in test case 6, 13, 19, 26 and 35 etc. Test case with the “result obtained” section highlighted in red indicates that the result is not as desired as the expected one. Explanation for these test cases are given in the following table:

<table>
<thead>
<tr>
<th>Test case(s)</th>
<th>Explanation for the undesired result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 3</td>
<td>The Login function is not implemented in this project because we do not want to complicate the prototype with Linux authentication programming.</td>
</tr>
<tr>
<td>26</td>
<td>The Nimrod/G application window will always pop up with the current released version of Nimrod/G because the command for creating a new experiment does not support background mode.</td>
</tr>
<tr>
<td>28 - 31</td>
<td>Links in the navigation frame will link the user directly to the respective page without any check of user authentication information. This is certainly due to the not-yet-deployed user login session control functionality. Again for a prototype system we need to concentrate on the main functionalities of the Portal.</td>
</tr>
<tr>
<td>33, 34, 37</td>
<td>These are small bugs of the prototype system. Can be fixed by adding some more error handling codes into the resource.jsp page. For a prototype system, bugs of such a scale are acceptable.</td>
</tr>
</tbody>
</table>

The testing of the prototype Portal indicates that the Portal has been credited with a certain level of error handling capability. This has in fact added to the system’s stability and usability.

5.6 Summary

This chapter has looked into the details of how the Nimrod/G Portal is developed. We first discussed some existing approaches to the implementation of web portals and after a feasibility study of those proposals we introduced our simple JavaBean/JSP solution. Along with the implementation we identified key components for the JavaBeans and how they are related to each other and to the JSP pages. With an overview perspective to the implementation, some example pieces of codes are also given to help understand what is happening behind the scene.

The Portal is tested against 37 test cases, of which it has almost passed all. The results of the testing show that the implementation of the Portal has been a success, i.e. there has been a solution for the problem specified in this project.
Chapter 6. Evaluation

6.1 Introduction

The evaluation of this project will consist of two parts: one is the evaluation of the Nimrod/G Portal that we have implemented and the other is the evaluation of the project itself as a whole. Criteria are provided for the evaluation of each part. At the end of this chapter we examine each minimum requirements point defined in the Objective Form to see if the project goal has been achieved.

6.2 Portal Evaluation

To evaluate the Portal system we need criteria that can be used to evaluate certain aspects of our Portal. These criteria should be from a software engineering perspective.

6.2.1 Criteria

We introduce the criteria of functionality, performance and interface for the evaluation of the Portal. Functionality concerns about the functional aspects of the implemented Portal. Performance requires the Portal to be both reliable and robust. Interface examine how friendly and how easy the Portal can be used.

6.2.2 Functionality

As defined in the end of section 4.2.3 there are four main modules that SHOULD be implemented for the Portal. We discuss how the implementation of these modules has been achieved:

1) Create New Experiment: section 5.4.2.2 details the implementation of this module page and section 5.4.1 describes functionalities of the related classes.
2) Monitor Experiment: section 5.4.2.3 has every the implementation details for this module and again section 5.4.1 describes how beans function.
3) Control experiment: implemented in same sections as 2).
4) Resource management: see section 5.4.2.4 and 5.4.1 for details.

For the system as a whole, the portal functionality is compared to the existing ones’ for evaluation, as shown in the following table:

<table>
<thead>
<tr>
<th>Aspects to compare</th>
<th>Nimrod/G Portal</th>
<th>NPACI HotPage</th>
<th>Nimrod Portal*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security login support</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support Parametric experiment</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Support Job Details information</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Support Job status control</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Provide server details and loads</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support QoS parameter control</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Experiment monitoring and control</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes</td>
</tr>
</tbody>
</table>
From the table we can see that some of the Grid Portal functionalities are not implemented in our Nimrod/G Portal due to various environment constraints. The Portal also sees its strong sides complementing the Nimrod Portal to have implemented the Job Details and Job Control modules. Furthermore, all under implemented functionalities will be considered as future work of this project except the **Plan File Editor** module, which we think is not important for the Portal.

### 6.2.3 Performance

The performance of the Portal can be approached in two ways: how it has improved the existing Nimrod client way for Grid computing and how reliable the Portal is.

Without the Portal, a user who wants to do an experiment will have to first use the Nimrod client tools to create a run file, use shell commands to run the experiment and then monitor experiment status and perform controls over it with the client GUI. These processes can take up significant amount of time, especially when a non-Linux user is doing so. The Nimrod/G GUI does not provide resource management function and the administrator needs to edit the gatekeeper file manually to add or remove nodes. With the Web Portal the administrator only needs to perform some mouse clicks to achieve the same goal. The Nimrod/G GUI tool also consumes much more CPU and memory resources than a Web-based thin client does to the user's machine. In short, the Web interface of Nimrod/G simplifies and ease operations to access Grid resources so a better performance is achieved.

Another way is to study the reliability of the Portal. We have tested the Portal with every possible use cases of it and achieved satisfying results, which means the Portal is reliable enough at this prototype stage of implementation. A considerable amount of implementation codes for the Portal has been for error handling, including the prevention of the user from inputting invalid experiment names and plan file texts etc. During the implementation of the Portal, exceptions from the JSP container were very carefully handled, with the aim to minimize the chance of seeing it again. The result is the Portal getting more and more robust and reliable.

### 6.2.4 Interface

The Web-based interface of the Portal has been trying to achieve better friendliness and usability through the implementation of the JSP pages as shown in section 5.4.2. The user interface has maintained a certain degree of common style when switching from one page to another. The page will notify the user with a red-colour warning message every time the user inputs something improperly. The pages also try to provide as much information as possible within the limited space. For example, in the Job Information section of the Experiment Monitoring page, total jobs number and ready/executing/done/failed jobs as well as their total numbers are presented in a managed manner to provide integrative information.

Colour scheme is also applied to certain sections in the pages to facilitate better delivery of contents. For instance, in the Plan File section of the Experiment Monitoring page, parameters and tasks are displayed in different colours. Files in the Experiment Files section are classified with different colours as well.
Although usability is not a primary concern for developing prototype portals, we believe the interfaces as designed in section 4.5 have successfully delivered some friendliness factors to the Portal user.

6.3 Project Evaluation

We evaluate the project as a whole in this section.

6.3.1 Criteria

We first look at objectives of the project to see how well they have been achieved. As for a second criterion we review what have been learned from this project, i.e. how much understanding has been gained. The third criterion is about management of time and resources through out the project.

6.3.2 Objectives

Objectives are the milestones for a project to achieve. We look back to the project objectives defined in the introduction chapter to see how well they have been achieved at this later stage of the project.

Objective 1. Understand Grid Computing. This objective has been achieved through background research on the Grid as we have demonstrated in the whole chapter 2.

Objective 2. Understand Grid Resource Management and Scheduling. Chapter 3 presents the evidence of the completion of this objective.

Objective 3. Study Nimrod/G. We have covered into every detail of Nimrod/G GRB in section 3.3 hence this objective is clearly achieved.

Objective 4. Design a portal for Nimrod/G. The whole Chapter 4 demonstrates a proposal solution for the design of the Portal. All design issues have been stated in this chapter so this objective has no doubt been achieved.

Objective 5. Implement the portal on a Grid testbed. The designed Portal has been implemented on node “testgrid3” of the Grid testbed in the school. Implementation issues are documented in Chapter 5, which is evident enough to indicate the completion of this objective.

Objective 6. Integrate the Portal and Nimrod/G. After the implementation of the Portal the integration process has actually been done. This is because the Portal Director is tightly coupled with the Nimrod/G Dispatcher and we just could not talk about Portal implementation without a concern of Nimrod/G. The success of the integration reflects achievement of this objective.

All six objectives set out at the beginning of the project have been met and the processes of meeting them are fully documented in the report. The project can be evaluated as a success from this point of view.

6.3.3 Learning

Learning is a most exciting factor in the process of a project. There is always a lot more to learn for a challenging project like this. Generally speaking what I have learned from this project include:
1) Grid knowledge. Before taking this project I have already known some basics of the Grid concepts. This project has given me the chance to have further developed my understanding of the Grid fundamentals featuring Grid Resource Management and Scheduling using GRBs.

2) Understanding of portal architecture and skills in designing and implementing them. Although the technologies I have used for portal implementation are simply JavaBean and JSP, they have still presented me a life cycle of implementing Web portals, for the fact that I did not have any experience working with Web-based applications.

3) Everything about Nimrod/G, especially its APIs.

6.3.4 Management

Time management for a project is a big challenge itself for me, as I have no experience working with degree project before. A project schedule was proposed and agreed with my supervisor at the early stage of the project, as shown in Figure 6.1.

<table>
<thead>
<tr>
<th>Mile Stone</th>
<th>Week 0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Week Ending</td>
<td>5/5</td>
<td>13/6</td>
<td>20/6</td>
<td>27/6</td>
<td>4/7</td>
<td>11/7</td>
<td>18/7</td>
<td>25/7</td>
<td>2/8</td>
<td>9/8</td>
<td>16/8</td>
<td>22/8</td>
<td>29/8</td>
</tr>
<tr>
<td>Research / literature review 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research / literature review 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed Report Structure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research / literature review 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements &amp; Tasks Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install &amp; Test Nimrod/G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design &amp; Implement Prototype Portal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration of Portal &amp; Nimrod/G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portal Testing &amp; Improvement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Report - first version</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Report - second version</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Report - Final version</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demonstration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Submission</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>02-Sep</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>04-Sep</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.1 Approved Project Schedule

The meanings of literature review 1 to 3 are:

- Research & literature review 1: Research on Grid Basics and Resource Management & Schedulers, initial study of Globus
- Research & literature review 2: Indepth study and research on Globus and NimRod
- Research & literature review 3: Indepth study and research on Grid Portal and Grid Resource access through portals

However, sticking to this schedule was later found out not an easy job. The literature review and background research were demanding more efforts than they had been planned and it was not until the 20s of July that I was able to start with the Portal Design. The same problem was found with the implementation of the portal – it was far more challenging than I had expected and lasted for another two weeks. The writing up of the design and implementation chapters, consequently, was delayed. The whole project was re-scheduled to reflect the actual works going every week, as illustrated in Figure 6.2.
6.4 Minimum Requirements

As a final method to evaluate the project, we shall compare the project achievements to the minimum requirements. Since the minimum requirements are objective 1 to 4 from the Project Objectives, and we have proved in section 6.3.2 that all project objectives are achieved successfully, we can conclude that all minimum requirements have been met.

6.5 Summary

We evaluated the implemented Portal and the project as a whole respectively in this Chapter. In summary the evaluation of this project has shown that project is a success both on the prototype Portal system and the project sides. The project has met all of its objectives and the Portal met its specifications, which is evidence enough of the project being a success.
Chapter 7. Conclusion

7.1 Introduction

This chapter brings together the achievements of the project, highlighting areas of particular success and looking at possible future work in the area of this project.

7.2 Project Achievements

The project looks into the world of Grid and studies the important issues in this field, with a particular attention to resource management and scheduling on the Grid. Major aspects of the Grid and solutions provided by the Globus Toolkit are examined in Chapter 2 and examples for GRBs and Grid portals are well studied in Chapter 3. In order to design the portal for Nimrod/G, the broker’s behaviour and APIs are fully examined in section 3.3.

The design and implementation of the Portal in this project has successfully presented a solution for the problem identified in Chapter 1. The following functionalities have been provided with the Portal to enable Web access to Nimrod/G:

- List all experiments
- Create, remove, control and monitor parametric experiments
- Manage resources for experiments

7.3 Possible Enhancements and Improvements

The developed Portal can be improved in several aspects. A major extension to this project would be a module to handle security issues on the Grid. The GSI service can be used to address this issue. The authentication should be at system level instead of at database level. Another under developed module that needs special attention is about the QoS control functionality. This can be achieved by utilizing the Nimrod 2 QoS APIs to set deadline/budget for experiments.

The Experiment Resource module can be enhanced by adding a new function to the ExperimentResource Bean to perform queries for detailed status of resource nodes. This can be done by invoking MDS services from the relevant bean to perform queries.

A database can to be set up as a future work for the project. User account information and user specific experiment information can be stored into database for better retrieving. Quick retrieving also adds to the system performance.

A final improvement to the Portal could be a Plan File Editor to help the user write up the plan file. The editor should provide guidance for writing up parameters and tasks. An example for this can be referenced from the Applet Plan File Editor developed by the Nimrod team for Nimrod Portal. The user should also be given the choice of editing the plan file prior to the start up of experiment.

7.4 Project Summary

The conclusion of this project has seen a solution to the design and implementation of a Web-based portal for the Grid Resource Broker of Nimrod/G. The implemented portal has been tested and evaluated, the results of which indicate that the project objectives are well met and that the project process has been a success.
Appendix A. Personal Reflection

My experience of doing this project may benefit students who have a non-science background but doing technically demanding project like I.

I have to admit that this is rather challenging a project for me. Before taking this project, I had planned to choose a technical one because I wanted to learn more technical skills. The mysterious nature of the Grid and the technical skills involved in the process of portal development drove me to take this project as my first choice. It was after the first meeting with Karim that I began to worry about the challenges: the areas of Grid resource management and scheduling, Nimrod/G, portal design and implementation are all new to me; what I was qualified for choosing this project had been my knowledge of general Grid concepts and skills in Java programming. After dealing with all the challenges I found some experiences worth sharing.

First thing I want to say is that do not hesitate to ask. There were times when I found problems that seemed not possible to solve but after a discussing with classmates there would always come up with solutions. During the design stage of the portal I was confused about Nimrod/G’s Director-Dispatcher mechanism and after several e-mails with the Nimrod team the solution became clear.

The other thing is about time management. Programming can be fun and giving us sense of achievement but the report is much more important. After the implementation of the portal I debugged the system for a week of time but later found my time for the report too pressed and I just needed to work over night for days to keep up with the project progress.

There were times when I just felt totally stressed and tired about coding or writing up report. My suggestion to cope with strains is firstly to keep quiet, because shouting will not add up anything to the project but worries; secondly to go for some sports and get refreshed.
Appendix B. MSc Project Marking Scheme, Interim Report Comments, Project Objectives & Deliverables

School of Computing, University of Leeds
MSc PROJECT INTERIM REPORT

All MSc students must submit an interim report on their project to the MSc project co-ordinator (Mrs A. Roberts) via CSO by Thursday 8th May 2003. Note that it may require two or three iterations to agree a suitable report with your supervisor, so you should let him/her have an initial draft well in advance of the deadline. The report should be a maximum of 10 pages long and be attached to this header sheet. It should include:

- the objectives, deliverables and agreed marking scheme
- resources required
- progress report and project schedule
- proposed research methods
- a draft chapter on the literature review and/or an evaluation of tools/techniques
- the WWW document link for the project log to date

The report will be commented upon by both the supervisor and the assessor in order to provide you with feedback on your approach and progress so far.

_The submission of this Interim Report is a pre-requisite for proceeding to the main phase of the project._

<table>
<thead>
<tr>
<th>Student:</th>
<th>Lin Ruofei</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programme of Study:</td>
<td>MSc Information Systems</td>
</tr>
<tr>
<td>Title of project:</td>
<td>Grid Resource Management and Scheduling</td>
</tr>
<tr>
<td>Supervisor:</td>
<td>Dr. Karim Djemame</td>
</tr>
<tr>
<td>External Company (if appropriate):</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**AGREED MARKING SCHEME**

<table>
<thead>
<tr>
<th>Understand the problem</th>
<th>Produce a solution *</th>
<th>Evaluation</th>
<th>Write-up</th>
<th>Appendix A</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>40</td>
<td>20</td>
<td>15</td>
<td>5</td>
</tr>
</tbody>
</table>

* This includes professionalism

Signature of student:  
Date: 08/05/03

_Supervisor's and Assessor's comments overleaf &_
Supervisor's comments on the Interim Report

This is a challenging project that brings together Grid computing, Grid portals, and Resource Management and scheduling. The project's objectives are clearly defined. However, the mid-project report has been written in a bit of a rush.

K.D.

Assessor's comments on the Interim Report

Seems very broad. Needs a good case study to make progress.

What is a good outcome? i.e. what will you demonstrate?

Comparison with other Portal and its needs?
Objectives

As agreed with Dr. Karim Djemame, this project will try to achieve the following goals:

3. Study Nimrod/G:
   a. Understand its declarative parametric modelling language for expressing a parametric experiment.
   b. Installation and testing to see how it works.
4. Design a portal for Nimrod/G.
5. Implement the portal on a Grid testbed.
6. Integration of portal and Nimrod/G.

Minimum Requirements:

Overall Objectives 1 to 4.

Deliverable(s):

1. A project report.
2. Grid software (a portal).

Software and Hardware Resources Required:

Grid testbed and Globus Tookit.
Appendix C. Use Case Realisation

This appendix shows the Use Case Realisation for each Use Cases (including sub Use Cases) of the Nimrod/G Portal system, using the UML Collaboration Diagrams.

1 Create New Experiment:

2 Monitor Experiment

2.1 View experiment status

2.2 View plan file

2.3 View experiment files

2.3.1 View all files

2.3.2 Download experiment file

**Note: step 7 upload() is optional for the user.**
2.4 View job information

2.4.1 View jobs

2.4.2 View job detail

2.4.3 Set Job Status

2.5 View resources

3 Control Experiment

3.1 Start experiment

3.2 Pause experiment

3.3 Resume experiment
3.4 Archive (remove) experiment

4 Manage Resources

4.1 Add node

4.2 Delete node
Appendix D. Design of the User Interface

Create New Experiment page

![Create New Experiment page diagram]

- **Quick links for Portal functions**
- **Navigation Frame**
- **Main Frame**
- **Plan file text area**
- **For file upload**
- **Select resources**
Monitor Experiment page

Quick links

Experiment status section

Links for experiment control

Job status section, clicking on a job for job details and job control

Show the plan file texts here

List of experiment files, click on a file to download it

List of resources being used

---

**Status of: experiment1**

The experiment has not yet started.

Click the link below to start.

Start Experiment  Archive Experiment

**Job Status**

There are 36 jobs for this experiment.

---

<table>
<thead>
<tr>
<th>Ready</th>
<th>Executing</th>
<th>Done</th>
<th>Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

---

**Plan File**

```plaintext
parameter x label "Time to sleep" integer select anyof 145 160 175 190 default 145 160 175 190;

parameter y label "Dummy1" text select anyof "a" "b" "c" default "a"

/*empty*/

task nodestart

    copy test.pla node1:
    endtask

    task main

    node:execute /nfs/sleep
    endtask
```

**Experiment Files**

- directory: a.org
  - experiment1.pla
  - experiment1.run
  - gatekeeper
  - nimrod.log
  - schedule.summ

**Resources**

<table>
<thead>
<tr>
<th>Type</th>
<th>Host (Node)</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>glnbus</td>
<td>grid2.leeds.ac.uk (s1)</td>
<td>ready</td>
</tr>
<tr>
<td>glnbus</td>
<td>testgrid2.leeds.ac.uk (s2)</td>
<td>busy</td>
</tr>
<tr>
<td>glnbus</td>
<td>testgrid3.leeds.ac.uk (s3)</td>
<td>down</td>
</tr>
<tr>
<td>glnbus</td>
<td>testgrid4.leeds.ac.uk (s4)</td>
<td>busy</td>
</tr>
<tr>
<td>glnbus</td>
<td>testgrid6.leeds.ac.uk (s5)</td>
<td>busy</td>
</tr>
</tbody>
</table>
Manage Resources page

Add node section

List of nodes for choice of deletion

Experiments page

List of experiments showing status and time elapsed. Clicking on an experiment leads to the Monitor Experiment page.
## Appendix E. Test Cases

<table>
<thead>
<tr>
<th>Test No</th>
<th>Area of Test</th>
<th>Test Case</th>
<th>Result Expected</th>
<th>Result Obtained</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Login Page</td>
<td>User clicks on Login button</td>
<td>Prompt user to input user name and password</td>
<td>User is directed to experiments page</td>
</tr>
<tr>
<td>2</td>
<td>Login Page</td>
<td>User inputs username and password and presses Login</td>
<td>Prompt user to input password</td>
<td>User is directed to experiments page</td>
</tr>
<tr>
<td>3</td>
<td>Login Page</td>
<td>User inputs username and password and presses Login</td>
<td>User account information authenticated in database and logged in to the experiments page showing user-specific experiments list</td>
<td>User is directed to experiments page showing all experiments list on the server</td>
</tr>
<tr>
<td>4</td>
<td>Experiments Page</td>
<td>User clicks on an experiment</td>
<td>User directed to the experiment monitor page</td>
<td>User directed to the experiment monitor page</td>
</tr>
<tr>
<td>5</td>
<td>Experiments Page</td>
<td>User clicks on create new experiment link</td>
<td>User directed to new experiment page</td>
<td>User directed to new experiment page</td>
</tr>
<tr>
<td>6</td>
<td>Experiment Monitor Page</td>
<td>User checks for experiment status</td>
<td>Experiment status correctly displayed on the page</td>
<td>Experiment status and time elapsed (if applicable) displayed on the page. Time elapsed is displayed in red.</td>
</tr>
<tr>
<td>7</td>
<td>Experiment Monitor Page</td>
<td>Experiment not yet started. User clicks on start experiment link to start the experiment</td>
<td>Experiment started with status changed to “executing”, pause experiment link and time elapsed shown on the page</td>
<td>Experiment started. The message of “The experiment is executing, you can pause it at any time” and pause experiment link and time elapsed are shown on the page.</td>
</tr>
<tr>
<td>8</td>
<td>Experiment Monitor Page</td>
<td>Experiment executing. User clicks on pause experiment link to pause the experiment</td>
<td>Experiment paused with status changed to “inactive” and resume link and time elapsed shown on the page</td>
<td>Experiment paused. If the experiment is terminating, prompt the user “the experiment is terminating, please wait”. If the experiment is inactive, prompt the user with “the experiment has started, but is currently paused”. Resume link and time elapsed are shown on the page.</td>
</tr>
<tr>
<td>9</td>
<td>Experiment Monitor Page (experiment status section)</td>
<td>Experiment paused. User clicks on resume link to resume execution of the experiment</td>
<td>Experiment restarted with status changed to “executing”, pause link and time elapsed shown on the page</td>
<td>Experiment restarted. The message of “The experiment is executing, you can pause it at any time” and pause experiment link and time elapsed are shown on the page</td>
</tr>
<tr>
<td>10</td>
<td>Experiment Monitor Page (experiment status section)</td>
<td>User clicks on archive experiment link</td>
<td>Experiment removed from the server</td>
<td>Prompt a confirm window to the user for confirmation of archiving</td>
</tr>
<tr>
<td>11</td>
<td>Confirm Window</td>
<td>User clicks “yes”</td>
<td>Experiment removed from the server</td>
<td>Experiment removed from the server and the user is directed to the Experiments Page.</td>
</tr>
<tr>
<td>12</td>
<td>Confirm Window</td>
<td>User clicks “cancel”</td>
<td>Nothing will happen, just close the confirm window</td>
<td>Confirm window closed and back to the experiment monitor page</td>
</tr>
<tr>
<td>13</td>
<td>Experiment Monitor Page (job information section)</td>
<td>User checks for job information</td>
<td>Job information displayed on the page</td>
<td>Job information on total number of jobs, ready/executing/done /failed jobs, and hyperlinked job Ids are displayed on the page</td>
</tr>
<tr>
<td>14</td>
<td>Experiment Monitor Page (job information section)</td>
<td>User clicks on a job ID link</td>
<td>Detailed job information displayed</td>
<td>A small new window popped up with detailed job information on it, including job status, server, task and parameters. Links for job control are also provided on that window.</td>
</tr>
<tr>
<td>15</td>
<td>Job details window</td>
<td>User clicks on set job done link</td>
<td>Job status set to “done”</td>
<td>Job status set to “done”, user gets prompted with “set job to done – OK!”</td>
</tr>
<tr>
<td>16</td>
<td>Job details window</td>
<td>User clicks on set job fail link</td>
<td>Job status set to “failed”</td>
<td>Job status set to “failed”, user gets prompted with “set job to fail – OK!”</td>
</tr>
<tr>
<td>17</td>
<td>Job details window</td>
<td>User clicks on set job wait link</td>
<td>Job status set to “ready”</td>
<td>Job status set to “ready”, user gets prompted with “set job to wait – OK!”</td>
</tr>
<tr>
<td>18</td>
<td>Experiment Monitor Page (plan file section)</td>
<td>User checks for experiment plan file</td>
<td>Experiment plan file text correctly displayed on the page</td>
<td>Experiment plan file texts correctly displayed on the page, with parameters</td>
</tr>
<tr>
<td></td>
<td>Experiment Monitor Page (experiment files section)</td>
<td>User checks for experiment files</td>
<td>Experiment files listed on the page</td>
<td>Experiment files listed on the page with system files shown in red and log files in green, user uploaded files and experiment output files are hyperlinked.</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------------------------</td>
<td>---------------------------------</td>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>20</td>
<td>Experiment Monitor Page (experiment files section)</td>
<td>User clicks on a file to download</td>
<td>Pop up the “Save file to..” window</td>
<td>Pop up the “Save file to..” window.</td>
</tr>
<tr>
<td>21</td>
<td>Experiment Monitor Page (resources section)</td>
<td>User checks for experiment resources</td>
<td>Experiment nodes listed</td>
<td>Experiment nodes listed on the page, with node type, node host, node server Id and node status provided into columns of a table</td>
</tr>
<tr>
<td>22</td>
<td>Create New Experiment Page</td>
<td>User forgot to enter any of experiment name, plan file or forgot to select nodes and click on the create button</td>
<td>Remind the user to input in the necessary fields</td>
<td>Prompt the user to enter valid experiment name, select at least one node, and remind the user “plan file missing”.</td>
</tr>
<tr>
<td>23</td>
<td>Create New Experiment Page</td>
<td>User inputs all fields but the experiment name already exists</td>
<td>Prompt the user to choose another name for the experiment</td>
<td>Prompt the user to choose another name for the experiment.</td>
</tr>
<tr>
<td>24</td>
<td>Create New Experiment Page</td>
<td>User inputs all fields and clicks on the “Create” button but the plan file is invalid</td>
<td>Prompt the user to check the plan file texts</td>
<td>Prompt the user to check the plan file texts.</td>
</tr>
<tr>
<td>25</td>
<td>Create New Experiment Page</td>
<td>User clicks “Browse” button</td>
<td>Pop up the browse file window for selection of file to upload</td>
<td>Pop up the browse file window for selection of file to upload.</td>
</tr>
<tr>
<td>26</td>
<td>Create New Experiment Page</td>
<td>User clicks “Create” with valid experiment name, plan file and at least one node selected</td>
<td>Create new experiment on the server with the specified name, plan file and resources</td>
<td>Create new experiment on the server with the specified name, plan file and resources. The Nimrod/G application window will pop up. Prompt user with “Created new experiment experimentName” with the experimentName hyperlink to the</td>
</tr>
<tr>
<td>Experiment Control Page</td>
<td>User clicks on the created new experiment name link</td>
<td>Direct user to that experiment’s monitor page</td>
<td>Direct user to that experiment’s monitor page</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Create New Experiment Page</td>
<td>User clicks on Experiments link</td>
<td>If user has been authenticated, display Experiments Page in the main frame, else prompt user the login page.</td>
<td>Show Experiments Page in the main frame.</td>
<td></td>
</tr>
<tr>
<td>Navigation Frame</td>
<td>User clicks the “New Experiment” link</td>
<td>If user has been authenticated, display Create New Experiments Page in the main frame, else prompt user the login page.</td>
<td>Show Create New Experiment Page in the main frame.</td>
<td></td>
</tr>
<tr>
<td>Navigation Frame</td>
<td>User clicks the “Resource Management” link</td>
<td>If user has been authenticated as administrator, display Resource Management in the main frame, else prompt user the login page.</td>
<td>Show Resource Management Page in the main frame.</td>
<td></td>
</tr>
<tr>
<td>Navigation Frame</td>
<td>User clicks the “Logout” link</td>
<td>Destroy the user session and display login page in the main frame</td>
<td>Show Login Page in the main frame.</td>
<td></td>
</tr>
<tr>
<td>Manage Resource Page</td>
<td>User clicks on the “Add Node” button without host address</td>
<td>Prompt user to input host address</td>
<td>Pop up message “Please input a valid node address!”</td>
<td></td>
</tr>
<tr>
<td>Manage Resource Page</td>
<td>User clicks on “Add Node” button with an invalid node address in the node name field</td>
<td>Prompt the user to input a valid node address</td>
<td>The invalid node host address is added to the node list</td>
<td></td>
</tr>
<tr>
<td>Manage Resource Page</td>
<td>User clicks on “Add Node” button with a valid node address in the</td>
<td>Prompt the user “node already exists”</td>
<td>The input node is added to the gatekeeper file regardless of the already existence of that node</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>node name field but the node name already exists in the gatekeeper file</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Manage Resource Page</td>
<td>User clicks on “Add Node” button with a valid node address which is not in the gatekeeper file in the node name field</td>
<td>Node added to the nodes list of the gatekeeper file</td>
<td>Node added to the nodes list of the gatekeeper file and the nodes list for deletion is updated and sorted with the new node in it</td>
</tr>
<tr>
<td>36</td>
<td>Manage Resource Page</td>
<td>User selects one node and clicks on the “Delete Node” button</td>
<td>Selected node deleted from the nodes list</td>
<td>Selected node deleted from the nodes list and the list is updated and sorted</td>
</tr>
<tr>
<td>37</td>
<td>Manage Resource Page</td>
<td>There is only one node left and the user selects it and click on “delete node” button</td>
<td>The final node is deleted from the nodes list and the delete node section should prompt the user with a message of “no node found in the gatekeeper file” and the “delete node” button should be disabled</td>
<td>The final node is deleted from the nodes list but when the server come to display the updated nodes list it throws an arrayIndexOutOfBoundsException exception.</td>
</tr>
</tbody>
</table>
Appendix F. Presentation Slides

Design and Implementation of a Web Portal for Nimrod/G

MSc Project Demo
Ruofei Lin (scs2rl)
MSc Information Systems
1 September, 2003

*Online Log: http://csiis.leeds.ac.uk/scs2rl/public/project.html

Project Objectives

- Understand Grid Computing
- Understand Grid Resource Management & Scheduling
- Study Nimrod/G
- Design & Implementation of a Portal for Nimrod/G

Sounds very broad!!
Project Objectives

- Understand Grid Computing
- Understand Grid Resource Management & Scheduling
- Study Nimrod/G
- Design & Implementation of a Portal for Nimrod/G

Concentrate on these!
Project Objectives

- Understand Grid Resource Management & Scheduling

GRB
Existing Broker Systems
Sun Grid Engine
Condor-G

Project Objectives

- Study Nimrod/G

Parametric language
Computational Economy
Nimrod APIs
Need for Web Portal
What I have done..

- Design
- Implement
- Testing
- Evaluation

**A Web Portal!!**

**Design**

- Make full use of Nimrod API commands
- (very) High Level  no Globus implementation
- Object-oriented
- Usability concerns
- Feasibility study working with constraints
Design

requirement

Design architecture

This page contains text and diagrams related to design and architecture. The diagrams illustrate various components and their interactions, which are detailed in the text section. The text discusses the design and architecture of grid resource management, including user interfaces and resource monitoring tools.
Design components

Implementation

No GPDK
No Struts
No GridShpere
No Perl
JUST JSP + JavaBean!
Implementation

JavaBeans

Implementation

JSPs
demo time ..

- Run experiment with Nimrod/G application
- Write up the plan file
- Write up the gatekeeper file
- Run experiment with Nimrod/G Portal
- Log in
- Create new exp
- Monitor/control experiment
- Manage resource nodes

Evaluation/Conclusion

Possible improvements
- Security issue manage user login
- Node status enquiry
  http://test.iri.leeds.ac.uk/ganglia@iri31
- QoS control need Nimrod 2
- Database to relieve Web server

Conclusion:
There has been a project-specific solution that meets the objectives fine.
References