g-Eclipse - An Integrated Framework to Access and Maintain Grid Resources

Harald Gjermundrød[#1], Marios D. Dikaiakos[#2], Mathias Stümpert[*3], Paweł Wolniewicz[^4], and Harald Kornmayer[&5]

[#] *University of Cyprus, PO Box 20537, 75 Kallipoleos Str., 1678 Nicosia, Cyprus*
[*] *Forschungszentrum Karlsruhe, Postfach 3640, 76021 Karlsruhe, Germany*
[^] *Poznan Supercomputing and Networking Center 61-704 Poznan, ul. Noskowskiego 10, Poland*
[&] *NEC Laboratories Europe, IT Division, Rathausallee 10, 53757 St. Augustin, Germany*

{[1] harald, [2] mdd}@cs.ucy.ac.cy, [3] mathias.stuempert@iwr.fzk.de,
[4] Pawel.Wolniewicz@man.poznan.pl, [5] harald.kornmayer@it.neclab.eu

Abstract

The g-Eclipse framework provides a general, integrated workbench toolset for Grid users, operators and developers. Based on the Open Source Eclipse ecosystem, g-Eclipse supports scientists to interact with Grid resources independent of the underlying Grid middleware. Its main objective is to deliver an extensible framework for different Grid actors, by providing a unified abstraction of the Grid. The Grid abstraction enables Grid application users to access the Grid in a desktop-like manner with wizards specific for common use cases; it also provides a set of visual configuration tools to maintain and configure Grid resources.

1. Introduction

In recent years, Grids have emerged as wide-scale distributed infrastructures that support the sharing of geographically distributed, heterogeneous computing and storage resources [16]. Grid middleware provides the abstraction to the user to interact with these resources. Several efforts are underway by both academia and industry to build these infrastructures and middlewares. Scientists have established numerous "Virtual Organizations" on top of Grid infrastructures to solve complex problems. Many Grid projects (such as EGEE [10], GridLab [12], DEISA [1], etc.) demonstrated the benefit of a general infrastructure for scientific and commercial applications. But even if there is a trend towards interoperable, service-oriented implementations of Grid-services, currently there are different Grid middleware systems in use. The most popular and widely distributed middleware systems are gLite [2], Globus Toolkit 4 [4], and UNICORE [15]. While all these middleware systems offer the basic services to interact with the underlying Grid infrastructure, each follows a slightly different approach.

Although the capabilities provided by Grid infrastructures have been demonstrated successfully in computational- and data-grid applications spanning across a wide spectrum of application areas [17, 18, 20], only a few Grid users use them extensively. Many of the potential users of the Grid restrain themselves from using the Grid because of the inherent complexity of Grid technologies. Understanding the behavior of Grid resources is difficult and the learning curve for newcomers is too steep. Therefore, more user-friendly and intuitive tools and user interfaces are needed to make the look-and-feel of Grid infrastructures similar to that of existing computer desktop systems which people are already familiar with.

In this paper, we first present the Grid abstraction that is provided by the g-Eclipse framework. The g-Eclipse project [11] uses the Eclipse platform to devise a middleware independent, integrated workbench toolset for Grid users, operators and developers. The current implementation supports the gLite middleware while support for the GRIA [5] middleware is underway. Second, we discuss two GUI-based perspectives for the Grid user and operator; these perspectives enable a scientist to run experiments on Grid resources and an administrator to manage the local site respectively.

The paper is structured as follows: Section 2 overviews Eclipse and the g-Eclipse architecture. Section 3 presents the abstraction of the Grid as supported by the g-Eclipse framework. Section 4 describes a number of tasks that a user or operator can perform using the g-Eclipse framework. Section 5 presents related work and Section 6 concludes.

2. Eclipse and the g-Eclipse Architecture

In this section we will first familiarize the reader with the Eclipse platform, that can be used as a basis to devise extensible frameworks. The g-Eclipse framework is such a framework, and its high-level architecture is presented next.

The Eclipse Platform

The Eclipse framework started as an integrated development environment for Java, and was later revised to be an open platform for a wide range of tools. The central point of the Eclipse architecture and framework is its plug-in architecture, a component-based software architecture that leads to a clear and modular design. Within the Eclipse framework everything is a plug-in that can be dynamically added to and removed from the running Eclipse instance.

In the Eclipse world, every plug-in amends the functionality of other plug-ins. This is achieved by the underlying OSGi framework that defines the dependencies between the different plug-ins, and how and when additional plug-ins are loaded [9]. In addition, OSGi defines the mechanisms of extension points and extensions. An extension point allows to plug-in new functionality based on an abstraction or definition to enhance existing functionality. For instance, plug-in `A` defines an extension point and the appropriate interfaces that need to be implemented by a third-party plug-in. Other plug-ins `B` and `C` follow this invitation and extend this extension point with new functionalities. This way of building software components leads to an extensible architecture with well-defined interfaces.

The g-Eclipse Architecture

The g-Eclipse framework provides extensions on top of the Eclipse platform that respect the common Eclipse guidelines. Thus, g-Eclipse contains separate views and editors for different functionalities. These functionalities are carried out by a set of tools, implemented as component-oriented Eclipse plug-ins that integrate seamlessly with other views and editors integrated in the three g-Eclipse Perspectives.

Due to a lack of space, we only present a part of the g-Eclipse architecture (which follows), which is depicted in Figure 1. The components presented are those that handle the authentication and resource access and usages. As can be seen from the figure, the g-Eclipse architecture comprises three levels: The bottom level is the Eclipse platform. The next level is a set of middleware independent plug-ins, which provide the abstraction of authentication to the Grid as well as how to access and use Grid resources. The third level is the middleware-specific plug-ins that implement interfaces specified in the middleware independent layer. The

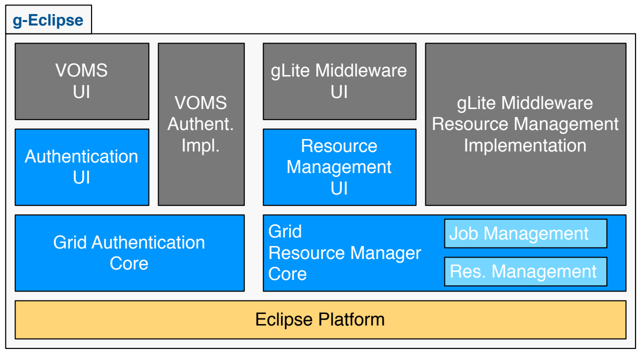


Figure 1. g-Eclipse architecture.

gLite middleware is shown as an example, but can be replaced by other middlewares.

3. The g-Eclipse Framework

The g-Eclipse framework presents an abstraction of the Grid to its actors along two dimensions. The first dimension is the role that the actor has with respect to the Grid. There are three roles that an actor can assume: the role of a Grid user, a Grid operator, and a Grid application developer. The second dimension is the level of expertise of the actor. The framework provides multiple interaction methods for using the Grid services, each with different levels of information and required input from the actor.

3.1 The g-Eclipse Grid Abstraction

The Grid is composed of loosely coupled heterogeneous services that the actor can access in a variety of interactive methods using various programming APIs. Complicating this even further is the non trivial task of locating the proper services and use their interaction methods.

g-Eclipse tries to present a coherent view of all these loosely-coupled services. This is accomplished via the higher-level abstractions of a *Virtual Organization* and a *Grid project*, which are supported by the user-interface of g-Eclipse. The former is a virtual community that an actor belongs to and thus can use the services it supports. The latter is a structure/placeholder where information about a Grid interaction sequence is stored.

The Virtual Organization

Some Grid middlewares, like gLite, organize their users and resources into Virtual Organizations (VO) in order to improve scalability and manageability. A Grid can group together its actors based on various metrics such as the actors' home country or region, whether or not an actor belongs to a specific experiment or based on the discipline of research of the specific actor. Of course, an actor can belong to multiple VOs. A site allocates its resources on a

VO basis instead of a per actor basis to improve scalability. Therefore, from a site's point of view, resources are supplied to VOs and not to individual actors. A Grid site specifies how its resources are allocated among the various VOs, and it also determines its local priority scheme, if any, among the VOs.

g-Eclipse supports the VO abstraction, which is a placeholder for an actor to configure the information required to use a VO that the actor belongs to. In order for an actor to use any of the resources belonging to a Grid, she first may have to authenticate herself using an authentication service, and then she may need an information service that lists all the resources supported by the specific VO. In this regard, the VO of g-Eclipse is a placeholder for the actor to enter the required information (end-points of authentication and information services) necessary to use the resources of a specific VO that the actor belongs to. This information is stored in the Eclipse workspace of the actor so it is only necessary to enter this information once.

The Grid Project

In the Eclipse world, an actor organizes her configuration and files into a Project. All the interactions that an actor initiates in a specific domain are also collected into this placeholder. In the Grid domain this is not the case; the actor has to organize, in an ad-hoc manner, its files and information that the actor uses for her various Grid tasks. It was envisioned that the information belonging to one research project should be organized into a Grid project. This project will present the actor with an abstraction of the Grid that is necessary to perform the work required for the specific scientific project. In this Grid project the actor specifies which VOs the actor wants to use for this scientific project. In addition, all the files, job descriptions, connections to Grid storage resources, information about submitted jobs, and so on are collected into one Grid project.

By providing this abstraction of the Grid in g-Eclipse it is envisioned that the productivity of the Grid scientist will improve. The Grid project can also be viewed as a holder for the current state of the actor's interactions with the Grid. In this case, the state can be stored and then later retrieved when the scientist wants to continue to work on the research project. As an added benefit of g-Eclipse being built on top of Eclipse, version control systems like SVN and CVS are readily available so that the Grid project can be checked in and out of these systems.

3.2 The Grid Perspectives

In order to simplify the usage of the Grid for various actor groups, g-Eclipse defined three perspectives to present a view of the Grid tailored to each of these actor groups. The groups identified so far are the Grid users, Grid operators, and Grid developers. The users are scientists that utilize the Grid to conduct their research by using its computational and storage resources. The operators administer the resources of local Grid sites. Finally, the Grid developers develop applications to be run on the Grid. This last perspective will only be presented in brief in this section (due to space limits) while the functionality of the other two perspectives will be presented in more detail in later sections.

The three different perspectives share some commonality and as Eclipse is a flexible platform the actor is free to define her own customized perspective. A perspective is a set of views, which are windows confined to present or represent a specific object or set of objects, editors, and multipage editors. The actor can re-arrange the views in the perspective as well as add and/or delete views.

Common Views

All the perspectives defined for g-Eclipse have some commonality as they all interact with the Grid. The Grid project that was presented above has its own view namely the *Grid Projects* view which looks like a file browser that the actor uses to browse through the items in the project (upper left in Figures 2 – 3). The *Properties* view is a standard view that comes with Eclipse. Its purpose is to be a common view for properties of a selected object in the other views (center bottom in Figures 2 – 3). The properties to be displayed and the amount of details are defined by the source object. An actor has to authenticate herself in order to access Grid resources, thus all the perspectives also have the *Authentication Tokens* view. This view contains a list of all the authentication tokens that the actor has created and they stay active until the application is closed; the tokens are maintained in memory only. The center space of the perspectives is used by the editors. As the actor opens files that are to be manipulated in an editor these are automatically opened in the center of the perspective.

User Perspective

The user perspective provides views for the actor to access the computational and storage resources provided by the Grid. The *Job* view (bottom center in Figure 2) lists the jobs that an actor has submitted to the Grid. In this view the actor can view the progress of the job and output files are linked to the job. The actor can also open the *Job Details* view that displays all the available information about the job that is selected in the *Job* view. The *Connection* view lists all the connections to local and remote storage resources that the actor has specified using the connection wizard. A multipage JSDL editor allows an actor to modify its job descriptions (see center of Figure 2).

Operator Perspective

The operator perspective provides views for an administrator to maintain her local Grid site. This perspective has an editor, the *Batch* editor (see center in Figure 3), which presents a graphical representation of the batch service of a local Grid site. As a service is selected in the editor, the batch jobs residing on that service are listed in the *Batch Job* view (center bottom in Figure 3). The *Glue Information* view conveys the static and dynamic information available from the information systems that the actor has configured in the VO setup (lower left in Figure 3). In addition, the perspective also provides the *Terminal* view where the administrator can SSH into the various machines to perform maintenance work.

Developer Perspective

The developer perspective will not be presented in this paper due to page limits. It assists Grid application developers in developing and debugging their applications (i.e. it is a tool to port applications to the Grid) [21].

3.3 Interaction Methods

As mentioned earlier, the second dimension that defines the Grid abstraction, is the expertise of the actor. Some actors will not need to use all the "bells and whistles" of the Grid, but want to be guided through the necessary steps in order to use the resources of the Grid. At the same time, others want to be able to have access to all the various options. To support the various levels of expertise of the actors, g-Eclipse provides both wizards and editors, in addition to cheat sheets, tutorials, and help pages.

Wizards

A wizard is a container for different pages, where each page guides an actor through the necessary steps to perform a task. By providing a set of wizards, g-Eclipse lowers the threshold to enter the world of Grids. These wizards will guide the actor through the first simple steps of using the Grid resources, like create a Grid project, specify a Grid job, and submit a job.

Editors

In Eclipse, an editor is more than what is normally referred to as an editor. In Eclipse an editor can be an ordinary document editor (text, xml, etc.) but it can also be an editor of a model, which manipulates that model graphically and textually. In addition, an editor can be a multipage editor in which the semantically similar items of a document are manipulated in their own page, hence the name multipage editor. g-Eclipse takes advantage of this powerful paradigm by providing editors to create and manipulate workflows, job descriptions, and even provide a graphical representation of a local Grid site that can be used to administer the local site.

Cheat Sheets, Tutorials, and Help Pages

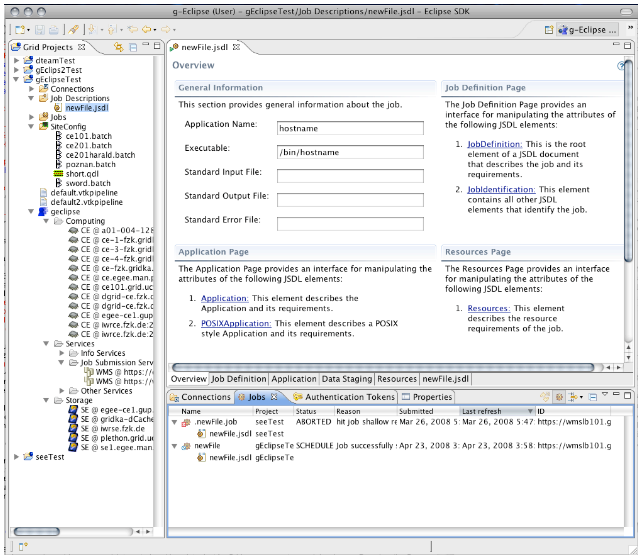
In addition to the actual wizards, editors, and views, Eclipse supports cheat sheets, tutorials, and help pages. All of these are integrated into the same workbench that the actor uses to interact with the Grid services. There is therefore no need to look for the help pages and tutorials on various web pages and switch between the Grid "tool" and the guidance "tools". The cheat sheets are active tutorials that guide the actors through the steps required to perform an action. If the actor struggles with a task she can command the cheat sheet to perform the action as the actor is observing.

4. User and Operator Perspective

All the g-Eclipse perspectives consist of a set of plug-ins that combined provide an abstraction of the Grid and assist the actor in the Grid interaction. This section presents a number of tasks that a user and operator can perform using the g-Eclipse framework.

4.1 User Interaction

The goal of the user perspective is to simplify the procedure for using Grid resources. This is accomplished by providing an intuitive GUI for a set of common Grid tasks that a user would normally perform. Using this GUI, a user is able to create a Grid project for a particular experiment (containing both data and jobs), use an editor to specify the job particulars, and gain access to various computational/storage resources.



General Information

This section provides general information about the job.

Application Name: hostname

Executable: /bin/hostname

Standard Input File:

Standard Output File:

Standard Error File:

Job Definition Page

The Job Definition Page provides an interface for manipulating the attributes of the following JSDL elements:

1. JobDefinition: This is the root element of a JSDL document that describes the job and its requirements.
2. JobIdentification: This element contains all other JSDL elements that identify the job.

Application Page

The Application Page provides an interface for manipulating the attributes of the following JSDL elements:

1. Application: This element describes the Application and its requirements.
2. POSIXApplication: This element describes a POSIX style Application and its requirements.

Resources Page

The Resources Page provides an interface for manipulating the attributes of the following JSDL elements:

1. Resources: This element describes the resource requirements of the job.

Name	Project	Status	Reason	Submitted	Last refresh	ID
.newFile.job	seeTest	ABORTED	hit job shallow re	Mar 26, 2008 5:	Mar 26, 2008 5:47:	https://wmslb101.g
newFile.jsdl	seeTest					
newFile	gEclipseTe	SCHEDULE	Job successfully :	Apr 23, 2008 3:	Apr 23, 2008 3:58:	https://wmslb101.g
newFile.jsdl	gEclipseTe					

Figure 2. User perspective.

Creating a Grid Project

To create a Grid project, the user simply selects to create a new *Grid Project*, which starts a wizard that guides the user through multiple pages. In the first page, the user specifies the name of the new project and its location within the filesystem. In the second page, the user is asked to select the VO that this project will "belong" to. If the user has not specified any VOs yet, she can provide the required info (contact URLs) to use a VO. Finally, the user can specify the directory structure she wants for her project, like a folder for *Job Descriptions* etc.

At the end of the Grid Project wizard, the user can view her project as shown in the upper left corner in Figure 2 (the Grid Project view). In addition to the directory structure that the user specified for the project, a virtual folder exists with the same name as the VO that this project is defined to use. In this folder the services that are available to the user from this VO are listed, like Computing, Storage, and services. The user can browse through these services to see what is available to her. As it will be explained in Section 4.2 this information is provided by the information service.

JSDL Multipage Editor

Job Submission Description Language (JSDL) is an open standard developed by the Open Grid Forum (OGF) [8] envisioned to be a middleware independent standard for specifying Grid jobs. Providing an open standard, it allows a user to take her job description and submit it to any middleware that supports the standard. In addition, there exist translators that translate a job description from JSDL to the description language supported by a specific middleware, i.e. a JSLD to JDL (Job Description Language) translator. JDL is the description language used by the gLite middleware.

g-Eclipse provides a multipage editor to specify and edit JSDL files (see center of Figure 2). A multipage editor assists the user in editing a document by grouping together common parts of the document in one tab and also provide instructions and information about the fields/properties/text that can be edited in that specific tab. Again, this provides a higher level of abstraction for the user compared to editing a pure xml file. In addition to simplifying the task of creating and editing a JSDL file, the JSDL multipage editor also helps in creating a well-formed-document with a structure that is verified. As a result, the user is given error messages when she has entered invalid data into fields.

The editor consists of six tabs with the following functionality:

Overview This tab presents information about what the user can/should specify in the other tabs (with hyperlinks). In addition, there are a few "main" fields like *Application Name* and the *Executable*.

Job Definition The user can set identifiers for the job, descriptions about the job, and the project that this job may belong to.

Application The user specifies information about the executable of the job. In addition the user can set properties like standard in/out/error files and environment variables.

Data Staging The user can specify where the files that the job needs are located and where the output should be placed.

Resource The user can specify requirements to the type of computational resource required for the job.

File Editor A syntax highlighted text editor is presented where the user can edit the XML document directly.

Access and Use of Computation Resources

Once the user has created a Grid Project and has specified one or more job descriptions, the task of submitting jobs to the Grid is simple. The user selects the job descriptions that she wants to submit to the Grid. Then, from the context menu of the selected jobs, *Submit Job...* is selected. This will launch a wizard where the user is presented with a list of available job submission services; the user selects one and submits the job.

After a job is submitted it will appear in the *Jobs* view (see center bottom in Figure 2). The user can observe the status of the submitted jobs. By clicking on a job, its properties are displayed in the *Properties* view (see center bottom in Figure 2). More details can be listed by opening the *Job Details* view. In this view the full history of the status changes, the computing element where the job is scheduled to execute and more are displayed. In case the job should not complete successfully, the user can use this view to track down where the job failed and why. After the job is successfully executed, the user can access the output files from the job through the links to the place on a storage resource where the files have been placed (as specified in the job's JSDL).

As can be seen from above, it is trivial for a user to access the computational resources; but under the "hood" the following steps are performed by a set of plug-ins to provide this simple abstraction. The `eu.geclipse.ui` plug-in contains the abstract wizard package `eu.geclipse.ui.wizards.jobsubmission`. The package (`eu.geclipse.glite`) implements this abstract wizard for the gLite middleware. The wizard gets the list of appropriate job submission services, which it retrieves from the `eu.geclipse.info`. gLite-middleware information is retrieved by the `eu.geclipse.glite.info` package, which fetches it from gLite's BDII information service specified in the Grid project. After the user has selected one of the job submission services, the wizard checks if the user has activated any Grid proxies, which are stored in `AuthenticationTokenManager` in the `eu.geclipse.core` plug-in. If no proxy is available, the `Authentication Token` wizard is launched. This wizard uses numerous (for the gLite middleware) plug-ins to contact a Virtual Organization Membership Service

(VOMS) server to create and activate a Grid proxy. With valid proxy (gLite middleware) the job is submitted to the Workload Management Service of gLite (WMS) using a secure connection. In order to monitor the status of the job in gLite there is again a set of plug-ins interacting with each other and with the LB (logging & bookkeeping) service.

Access and Use of Storage Resources

In addition to providing easy and intuitive access to computational resources, g-Eclipse also strives to provide access to storage resources in a similar fashion. The user of Grid resources would prefer to access the files in a similar way that she uses on her personal computer. The Grid has a replica management layer which controls access to files and data. Middlewares use different storage managers or even support more than one. Different access methods and protocols are used in order to gain access to these resources and in order to modify the content stored on these resources. g-Eclipse provides the user with a consistent ‘look and feel’ to access an abstract storage space. This provides for a unified view to different types of storage such as Storage Elements, GridFTP, DPM implementation of GridFTP or Storage Resource Manager (SRM) resources and local media, such as hard disk and removable media (CD, DVD, USB memory sticks). In addition to the support for Grid file systems, g-Eclipse uses the Eclipse File System (EFS) API and, therefore, any file system which was developed to work with the Eclipse Platform will automatically work with g-Eclipse.

The Grid is composed of distributed, heterogenous resources. Transferring data between different nodes involves the use of several tools. In g-Eclipse, these tools are selected automatically. Scientists and engineers can focus on their task rather than on which tools to use. The g-Eclipse data management subsystem hides low level information about file systems. The user deals with high level operations only – create directory, copy file, remove file, copy directory etc. Common user actions such as drag & drop or copy-paste are supported as well. File transfers that would take a long time can be put into the background to avoid blocking the user interface. Except for a time delay, user should notice no difference between remote and local file systems.

Ease of use is the highest priority for g-Eclipse. Earlier, users had not only to remember a lot of information but also had to remember where to get that information from. g-Eclipse utilizes the information system (see Section 4.2) in the background, so the user is able to browse the Grid and its storage elements without knowing anything about its location or contact strings.

Similar to the process of submitting a job to use the computational resources, there are a set of middleware independent and middleware specific plug-ins that interact with numerous Grid services, to provide to the end-user a simple and intuitive way of accessing Grid storage resources.

4.2. Operator Interaction

The goal of the operator perspective is to simplify the procedure of monitoring and maintaining a local Grid site. This is accomplished by providing an intuitive GUI for a set of common tasks that an administrator would normally perform. Using this GUI, an administrator can view the current state of the local Grid site and initiate commands from the context menu of the various resources to change the state of the resources.

Glue Information View

Knowledge of the actual status of Grid resources is essential for the operation and maintenance of a Grid with distributed resources. Thus, the Grid operator perspective provides plug-ins to access infrastructure information in a simple and interactive way. In addition to presenting the information to the user, it also provides an interface for other plug-ins to use this information. The information is collected from Grid information services, like the gLite BDII service, and cached locally. The information is organized in a data structure that resembles the Glue-schema v1.3 [13] and stored in a common data store so that it can be easily used by the rest of the plug-ins. Which information service to contact and its contact string is configured only once during the configuration of each of the VO that the user/administrator belongs to.

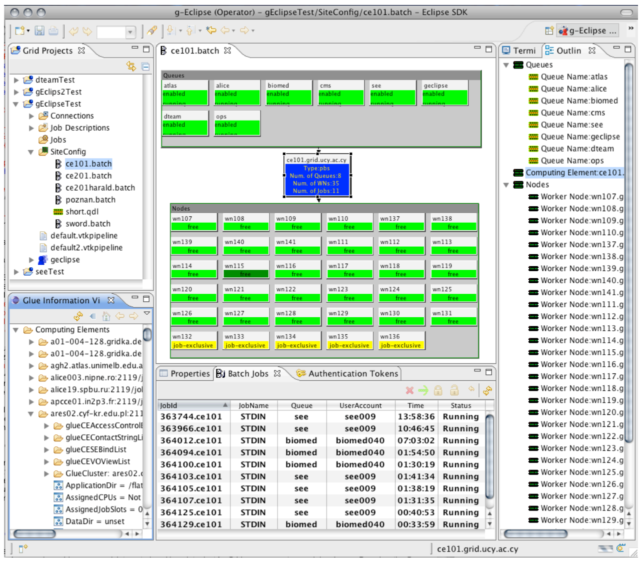
From the Glue information view, an administrator can browse the static and dynamic information published by her site, as well as other sites. In case the administrator has problems with her site, she can browse through the information published by her site to look for anomalies in the published information which can provide an indication as to which resource or service might be the cause of the current problem. The administrator can also compare values published by her site compared to what similar sites publish, in order to narrow down the cause of the problem.

Local Grid Site Monitoring and Configuration

The monitoring and configuration functionality are provided for batch-based Grid middleware systems. Multiple batch services exist like PBS, SGE, LSF and a batch service based Grid middleware supports one or more of these batch service implementations. The batch service is basically the traffic warden for the jobs that are to be executed on a Grid site. The jobs are placed into a queue where they wait until it is their time to run on one or more of the computational resources in the site. As the job executes the batch service polices its progress; after the job has completed, the service performs some clean-up and reports on a successful execution.

To improve manageability and scalability in batch-based Grids, the users are grouped. The local Grid site sets its policies of how to allocate its resources on a VO-basis rather than on an individual-user-basis. Jobs are placed into queues depending on job property like duration, resource requirements, and the VO the user that submitted the job belongs to. The site policy configuration may also influence in which queue the job is placed, i.e. if the duration of the job is less than 30 minutes then it may be placed in the "short" queue. The configuration of the queues is therefore of paramount importance to implement the resource sharing policy of a site. In addition, the policy of resource allocation may also change over time due to data challenges, requests from user groups, or organization policy changes. Therefore, a local Grid site administrator frequently needs to modify the setting of existing queues or delete them in addition to adding new queues.

In g-Eclipse, we have implemented an abstraction of the local batch service to convey the current state of the local Grid site to the administrator as well as to allow the administrator to modify the setting of the batch service. A batch service wrapper interface has been devised and developed. This interface provides for the functionality of gathering the current state of the batch service and also issues commands to modify the current setting of the service. If an implementation for this interface is provided for a specific batch service type then all the functionality presented below will be provided.



JobId	JobName	Queue	UserAccount	Time	Status
363744.ce101	STDIN	see	see009	13:58:36	Running
363966.ce101	STDIN	see	see009	10:46:45	Running
364012.ce101	STDIN	biomed	biomed040	07:03:02	Running
364094.ce101	STDIN	biomed	biomed040	01:54:50	Running
364100.ce101	STDIN	biomed	biomed040	01:30:19	Running
364103.ce101	STDIN	see	see009	01:41:34	Running
364105.ce101	STDIN	see	see009	01:38:19	Running
364107.ce101	STDIN	see	see009	01:31:35	Running
364125.ce101	STDIN	see	see009	00:40:53	Running
364129.ce101	STDIN	biomed	biomed040	00:33:59	Running

Figure 3. Operator perspective.

The batch service editor serves two purposes: depict in an intuitive way the current state of the available resources and allow to modify the setting of the resources. Figure 3 illustrates a screen-scrape of the operator perspective. The batch editor is in the center and it shows the resources (Queues, Computing Element, Worker Nodes) of the local Grid site in color code depending on their operational state. For large sites it is also possible to zoom out as to fit all the resources within the view. If the editor gets too cluttered all the resources are also listed in the outline view (left in the figure). By selecting a resource either in the editor or in the outline view, the properties of this resource are displayed in the the Properties view (bottom). In addition if there are any jobs currently residing on the selected resource they are also listed in the Batch Jobs view. From the context menu of the various resources, options to modify the setting of the selected resource(s) are given.

The operations explained below are usually performed by either typing commands in a terminal or by executing scripts that the administrator has devised for her site. g-Eclipse provides an intuitive GUI to perform these operations, hence simplifying and reducing the risk of executing erroneous commands.

Queue Operations The state of a Queue is determined by the state of two services, hence it has four states. The first service determines if the queue should accept any new jobs, so the queue can be enabled or disabled. The second service determines if the queue is transferring jobs from the queue to be executed on a working node so it is either running or stopped. From the context menu of one or more selected queue(s), the option is given to change the state depending on the current state. In addition if a selected queue is stopped, disabled and contains no batch jobs then the option to delete this queue is also given.

Computing Element Operations From the editor the only option available from the context menu is to create a new Queue. By selecting this option a wizard is launched. In the fist page of the wizard the required properties of the new queue are specified while on the second page optional properties can be added if that is desirable. A new queue could be created in response to a policy change or a data challenge. For instance a queue could be created that accepts short jobs with high priority for a subset of VOs.

Worker Node Operations A worker node can be either enabled or disabled and when it is enabled it may have jobs executing on it. An administrator can select one or more worker nodes and if they are all enabled then from the context menu the option to disable all of them is given. If they are all disabled the option to enable them is given in the context menu.

Batch Service Job Operations In the bottom in Figure 3 is the Batch Jobs view, which lists all the jobs that are residing on a selected resource in the batch editor. From the context menu of each of the jobs, the administrator has the option of deleting that specific job. In addition, if the job has not started to execute (queued) the option to move the job to another queue and/or computing element is given. By activating the move options a wizard is launched where the administrator will specify the new queue and/or computing element. An example of when a administrator would like to delete a job is if the job somehow misbehaves or something went wrong and the job "refuses" to finish. Similarly a site might want to try to move all its queued jobs before it performs an emergency shutdown.

5. Related Work

The GT4IDE project [7] and Globus Development Tools for Eclipse (GDTE) project [3] provide Eclipse plug-ins to access Globus-based Grids through Eclipse. This is in comparison to g-Eclipse which is a middleware independent framework which is accepted as a technology project by the Eclipse foundation, i.e. is the official Eclipse Grid project.

There exist numerous Grid ports like GENIUS [14], Sakai [19], and GridSphere [6]. Grid portals are Web-based facilities that provide a personalized, single point of access to Grid resources to assist the user in Grid related tasks. Portals are currently tied to a specific middleware and provide customized view for specific tasks that may be tied to a particular problem domain. g-Eclipse on the other hand provides a middleware independent, application domain independent, and a free standing pure Java Grid client to interact with Grid resources. In this sense g-Eclipse is more customizable and applicable to a larger audience since it is a tool for Grid users, operators, and developers.

6. Conclusions

In this paper we presented g-Eclipse, a framework for enabling various Grid actors to access Grid resources in an intuitive and easy way, i.e. by providing high-level abstraction of the Grid. g-Eclipse relies on the extension point mechanism of the Eclipse framework and is designed to be a Grid middleware-independent framework. It is the g-Eclipse vision that if the threshold of using the vast Grid resources is lowered, then more scientists will use them to further the state-of-the art in their respective discipline. In addition, as the task of providing Grid resources is simplified more institutions will be willing to share their resources.

Acknowledgements

This work was supported in part by the EU under projects g-Eclipse (#FP6-2005-IST-034327), CoreGRID (#IST-2002-004265), and the Eclipse foundation. We would also like to thank the member institutions of the g-Eclipse consortium and all the project members.

References

[1] Distributed European Infrastructure for Supercomputing Applications (DEISA). http://www.deisa.org/.

[2] gLite: Lightweight Middleware for Grid Computing. http://glite.web.cern.ch/glite/.

[3] Globus Development Tools for Eclipse. http://dev.globus.org/wiki/Incubator/GDTE.

[4] Globus project. http://www.globus.org.

[5] GRIA: Service Oriented Collaborations for Industry and Commerce. http://www.gria.org/ (accessed April 2008).

[6] GridSphere Portal Framework. http://www.gridsphere.org/gridsphere/gridsphere.

[7] GT4IDE: Globus Toolkit 4 IDE. http://gsbt.sourceforge.net/content/view/12/29/.

[8] Job Submission Description Language Working Group (JSDL-WG). Open Grid Forum. http://forge.gridforum.org/projects/jsdl-wg (accessed Apr. 2008).

[9] OSGi Alliance. http://www.osgi.org/.

[10] The Enabling Grids for E-sciencE Project. http://www.eu-egee.org/.

[11] The g-Eclipse Project. http://www.geclipse.eu/.

[12] The GridLab Project. http://www.gridlab.org/.

[13] S. Andreozzi et al. GLUE Schema Specification, version 1.3. http://glueschema.forge.cnaf.infn.it/Spec/V13/ (accessed Apr. 2008).

[14] R. Barbera. The GENIUS Grid Portal. In *Computing in High Energy and Nuclear Physics*, La Jolla, California, March 2003.

[15] D. W. Erwin and D. F. Snelling. UNICORE: A Grid Computing Environment. In *Lecture Notes in Computer Science*, volume 2150, pages 825–834. Springer, 2001.

[16] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*, chapter 4: Concepts and Architecture, pages 37–64. Elsevier, 2004.

[17] I. Foster and C. Kesselman. Scaling System-Level Science: Scientific Exploration and IT Implications. *Computer*, 39(11):31–39, November 2006.

[18] F. Gagliardi, B. Jones, M. Reale, and S. Burke. European DataGrid project: Experiences of deploying a large scale testbed for E-science applications. *Lecture Notes in Computer Science*, 2459:480–499, 2002.

[19] C. Severance, J. Hardin, G. Golden, R. Crouchley, A. Fish, T. Finholt, B. Kirschner, J. Eng, and R. Allan. Using the Sakai collaborative toolkit in e-Research applications. *Concurrency and Computation: Practice & Experience*, 19(12):1643 – 1652, August 2007.

[20] P. M. Sloot, A. Tiramo-Ramos, I. Altintas, M. Bubak, and C. Boucher. From Molecule to Man: Decision Support in Individualized e-Health. *Computer*, 39(11):40–47, November 2006.

[21] P. Wolniewicz, N. Meyer, M. Stroinski, M. Stuempert, H. Kornmayer, M. Polak, and H. Gjermundrod. Accessing Grid Computing Resources with g-Eclipse Platform. *Computational Methods in Science and Technology*, 13(2):131–141, December 2007.