A Middleware Independent Grid Workflow Builder for Scientific Applications

David Johnson
School of Biological Sciences
Lyle Building
University of Reading
Whiteknights
Reading, RG6 6BX, UK
d.johnson@reading.ac.uk

Ken Meacham
IT Innovation Centre
2 Venture Road
Southampton
SO16 7NP
UK
kem@it-innovation.soton.ac.uk

Harald Kornmayer
NEC Laboratories Europe, IT Research Division
Rathausallee 10
53757 St. Augustin
Germany
harald.kornmayer@it.neclab.eu

Abstract

Grid workflow authoring tools are typically specific to particular workflow engines built into Grid middleware, or are application specific and are designed to interact with specific software implementations. g-Eclipse is a middleware independent Grid workbench that aims to provide a unified abstraction of the Grid and includes a Grid workflow builder to allow users to author and deploy workflows to the Grid. This paper describes the g-Eclipse Workflow Builder and its implementations for two Grid middlewares, gLite and GRIA, and a case study utilizing the Workflow Builder in a Grid user's scientific workflow deployment.

1. Introduction

Workflow composition is an important part of reusing existing scientific analysis methods. Many of these scientific methods require high-performance computing (HPC) resources, and these are commonly provided by large clusters and supercomputers, or through using distributed commodity hardware forming computational grids. Tools for connecting to and manipulating such resources are usually middleware specific with no universal standard for HPC or Grid computing. Likewise, tools for workflow authoring already exist, but they are not designed to cater for multiple middlewares. In this paper we describe the Workflow Builder plugin of the g-Eclipse [1, 2] workbench, a universal graphical user interface for accessing existing Grid infrastructures, and its exemplary support for the authoring and submitting of workflows to two different middlewares, gLite and GRIA. We provide an overview of the different functionality that is available to a Grid User actor and how the Workflow Builder integrates with the job and data management tools in g-Eclipse. Finally we report on how g-Eclipse was used to demonstrate scientific and industrial applications with the Workflow Builder being a core component of a Grid User's activities.

2 Background

As the need for processing, analyzing and simulating large sets of data grows, so does the need for larger computing and data capacity. The field of Grid computing addresses this by providing methods for multiple collaborating entities to share distributed compute resources under an umbrella of a *virtual organization* [3]. For example, one of the aims of the EGEE project is to provide the worldwide scientific research community with a universally accessible Grid infrastructure. This infrastructure is divided into a number of virtual organizations segmenting groups of users, resources and institutions that map onto logical administrative domains. Grid middleware is software that is used as the 'glue' that 'sticks' together distributed computing and storage resources, presenting a standardized interface to the collective resource, where from a user-perspective, the underlying disparity of resources is hidden. The EGEE

project developed the gLite middleware and included components taken from a number of Grid projects [4] including Datagrid (EDG), DataTag (EDT), DataGrid (EDG), INFN-GRID, Globus and Condor. Although the gLite middleware goes some distance to simplify the underlying organization and management of Grid computing resources, interfacing with gLite-based virtual organizations is typically carried out through command-line sessions and interacting with the different subsystem and services individually. Some effort has been made to create graphical clients to access EGEE's Grid infrastructure, however these approaches targeted specific functionality such as job submission or data management, or produced domain-specific user interfaces. There was no single graphical client software that encapsulates Grid user functionality available for interfacing with EGEE Grids, and this is atypical of many other Grid and high-performance computing (HPC) infrastructures.

The European Commission Sixth Framework funded *g-Eclipse* project built an integrated workbench framework to access and operate existing Grid infrastructures and is described in detail by Gjermundrød et al [5]. Based on top of the open-source Eclipse framework, the g-Eclipse workbench provides tools to customize Grid users' applications, to manage Grid resources, and to support the development cycle of new Grid applications. The project aimed at providing generic Grid workbench tools that can be extended for many different Grid middlewares, such as gLite, UNICORE [6], and Globus [7], and it comes with exemplary support for the gLite and GRIA [8, 9] middlewares. An adapter to Amazon's Web services, Simple Storage Service (S3) and Elastic Compute Cloud (EC2) [10], was also developed which introduced Cloud Computing support to what was initially a project intended for just the Grid.

An integral part of the g-Eclipse workbench is the Workflow Builder plugin that allows users to compose workflows made up of Grid job descriptions. These job descriptions describe the semantics of a Grid job including executable definitions and locations of input and output data. The Workflow Builder enables users to take these descriptions and stitch them together into a workflow. These workflows can then be executed or deployed using the standard job submission mechanism provided by the g-Eclipse workbench (which is also used for executing single jobs) where conversion to middleware specific workflow descriptions is done on-the-fly.

3 The g-Eclipse Workflow Builder

The problem of expressing Grid workflows has been addressed by many different projects, however there has not been any single adopted standard for expressing and persisting workflows. While the g-Eclipse Workflow Builder attempts to cater for different kinds of Grid middleware systems that support workflows, it must be noted that the model is not intended to be a solution to enable interoperability between workflow systems. The approach taken is to provide a workflow builder that fits into g-Eclipse's own generalized Grid model abstractions whilst being able to cater for other middlewares.

The g-Eclipse Grid model abstracts a compute job to be executed as a *Job Description*, and a job that has been submitted to execute on the Grid as a *Job*. The workflow model only links into the Job Description abstractions, as the Workflow Builder plugin does not yet link into the job monitoring functionality that is present in the workbench. Job Descriptions provide a standard structure in which to define the parameters of a Grid Job and typically includes the location of an executable, locations of input data and locations on the Grid to write output data to. In g-Eclipse, Job Descriptions are expressed in an Open Grid Forum's standard, the Job Submission Definition Language (JSDL) [11], and Job Descriptions are saved in the workbench file system as JSDL files. When creating a new Job Description, a wizard provides a step-by-step interface for users to populate the most important fields in a new JSDL file and g-Eclipse includes a multipage JSDL editor (see figure 1) in which users can edit a variety of parameters defined by the standard.

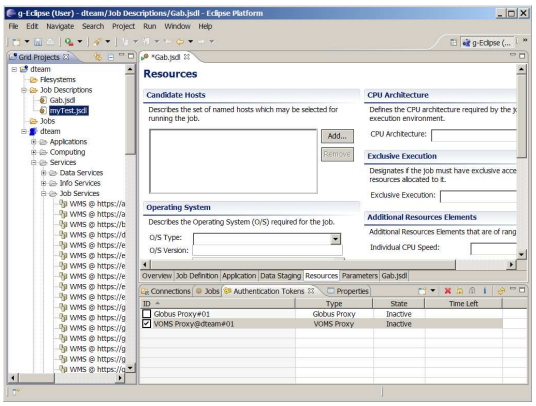

Figure 1. The g-Eclipse workbench with the JSDL Editor displayed in the main view.

Workflows can be thought of as process-oriented systems, where an overall process is broken down into sub-processes [12]. These sub-processes can then be organized in such a way as to optimize their execution, for example executing processes that are not dependent on each other in parallel to reduce total execution time. A workflow describes semantic relationships between sub-processes. In Grid workflow systems these relationships are almost always data flow dependencies, where the input of a given sub-process depends on one or more outputs of another.

To maintain consistency with the rest of the Grid model, workflows are modeled in g-Eclipse as a set of Job Descriptions (as JSDLs) and the data dependencies between them. This allows users to reuse existing single Job Descriptions to build more complex processes.

The Workflow Builder presents users with a graphical diagramming canvas on which to build their workflow. A palette allows users to pick and drop four kinds of element on to elements of the diagram. The following list of elements are illustrated in figure 2:

- *Workflow Job* - A container for Job Descriptions illustrated as a rectangle on the diagram canvas. By default, adding a Workflow Job to a diagram logically adds an empty description. As such, a context menu action or double-clicking on an empty Workflow Job loads up the JSDL wizard so that the user can define a new Job Description and associate it with the Workflow Job.
- *Input Port* - A representation of a single location of data intended as input for a Workflow Job and illustrated as a small square with a downward facing white arrowhead placed on a Workflow Job. Any number of Input Ports can be added to Workflow Jobs. Double-clicking on a Input Port allows a user to provide a URI location of some associated data.
- *Output Port* - Similar to an Input Port, however represents output data locations but is decorated with an upward facing white arrowhead. Adding and removing ports updates the associated Workflow Job's JSDL file's corresponding data-staging properties.
- *Link* - Representation of the actual flow between Workflow Jobs illustrated as an arrow between Input Ports and Output Ports. Links can only flow out of an Output Port and in to an Input Port.

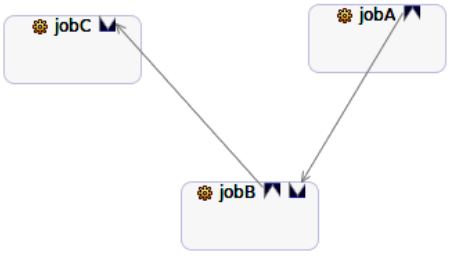

Figure 2. Three jobs with ports and connected by links.

Although a workflow can be built from scratch using the palette, a degree of automation has been implemented into the g-Eclipse Workflow Builder. Users can take their existing JSDLs that they may have authored using the JSDL Editor, or imported from elsewhere, and can quickly use them to build a new workflow. JSDL files that are present within a g-Eclipse Grid project explorer can be drag-and-dropped directly onto a workflow diagram canvas. When a JSDL file is dropped on the canvas, a new Workflow Job is created, and corresponding Input Ports and Output Ports created according to the job description. A context menu action has also been implemented to attempt to automate the creation of links between ports. When the "auto-connect" action is executed, the Workflow Builder attempts to determine the semantic links between Input Ports and Output Ports by searching for what URI locations match each other. For example, where an Output Port's URI matches an Input Port's, a link is automatically added.

A Grid workflow in g-Eclipse is saved as a combination of files. When JSDLs are added to a workflow diagram, a copy of the original JSDL is created and uniquely associated with the diagram. This was done to ensure that JSDLs could not be modified and moved outside of the context of the workflow once they have been associated with a Workflow Job. The actual description of the graphical diagram and its semantics is saved in a file in XML Metadata Interchange (XMI) format [13], which is an XML-based serialization of the graphical model defined by the Eclipse Graphical Modeling Framework (GMF) [1]. When inspecting a workflow diagram project, a single diagram is seen as a file-system directory containing the XMI description and the associated JSDL files.

4 Supported Middlewares

Workflows are handled by different middlewares in different ways, and the two exemplary middleware implementations for g-Eclipse highlight this. In g-Eclipse, a user can use the context menu on a workflow in the Grid Project explorer to execute middleware-specific functionality that is provided by the corresponding middleware plugins where appropriate. One must remember that g-Eclipse does not provide any form of workflow orchestration or enactment but acts only as an authoring tool and delegates workflow descriptions to the Grid middleware to handle. At the time of writing, the g-Eclipse Workflow Builder provides implementations for two major Grid middlewares, gLite and GRIA.

[1] Eclipse GMF was used as an aid to develop the graphical model that the Workflow Builder is based on. More details on GMF can be found at http://www.eclipse.org/gmf

4.1 gLite

The gLite middleware provides a user-centric workflow engine that allows any Grid user to try and submit a workflow to the Grid. Bascially, gLite handles workflow jobs in the same way as it does single jobs. gLite does not accept JSDL files, so the g-Eclipse job submission mechanism translates the JSDL descriptions on-the-fly into JDL (Job Description Language, developed for gLite specifically), where the process is transparent to the user. JDL supports its own description of workflows in a special case of JDL file. Regular JDL files describe a single anonymous job. Workflow JDL files contain multiple jobs which are labelled, and an extra attribute that describes the dependencies as a set of label tuples is included. In gLite the dependencies must describe a direct acyclic graph (i.e. the dependency graph must have no cyclic dependencies), and such workflow jobs are referred to as DAGs.

For gLite workflows, it must be noted that the intended use-cases for workflows make several assumptions. Firstly, each defined job in a workflow must be valid in that an executable application is available, and the input and output data locations are valid and reachable. Second, the required resources are available to the user. Finally, the individual jobs have been configured to deal with long delays on the Grid. In our experience, we found that when testing workflows against the EGEE infrastructure using gLite, resources are not always immediately available. As a result, the sub-jobs within a workflow can easily timeout or reach their maximum retry count before the workflow completes. Although this is a difficulty we experienced in our development tests, once the workflows are submitted to gLite from g-Eclipse, the responsibility for its execution lies with the middleware.

4.2 GRIA

The GRIA middleware differs from gLite in that it only allows users to execute applications that have already been installed on a GRIA Job Service, by a service operator. This means that typical Grid users are restricted to specific parameters afforded by those applications, and g-Eclipse handles this by auto-populating a JSDL when a user selects a particular application returned by the GRIA Job Service in the JSDL Wizard. The workflow functionality in GRIA is also not treated in the same manner as in gLite. GRIA workflows are not generally intended to be user-centric, but for Grid operators to use to provide encapsulated complex services for users.

However, GRIA provides application service packages to support either workflows published on a GRIA Job Service (and consumed by end users as GRIA applications) or an additional package to add support for the user submitting a workflow to be published automatically by the GRIA service. The GRIA Workflow Application [9] provides a command-line tool which takes a workflow description expressed in XML Simple Conceptual Unified Flow Language (XScufl) [14], and a few other parameters, and creates a new application on the GRIA Job Service that exposes the workflow to users as a single encapsulated process. The GRIA Workflow Deployer Application can also be installed on a GRIA Job Service (as a GRIA application); this wraps the command-line workflow deployment tool, taking in an XScufl workflow as input, as provided by the end user. The result is a newly deployed GRIA application that implements the workflow.

g-Eclipse authored workflows can be translated into XScufl descriptions using a context menu action. After translation, the GRIA Workflow Deployer Application can be accessed by a user via the standard g-Eclipse job submission mechanism (by creating a new JSDL to use the GRIA Workflow Deployer Application) to upload and install the new workflow to the Job Service site, whereby a new application becomes available to the user that encapsulates the new workflow's functionality. The user can then execute the workflow by selecting the new application using the JSDL Wizard and submitting the JSDL to the GRIA Job Service.

5 Case Study: A Pharmaceutical Application

To illustrate the use of the g-Eclipse Workflow Builder a number of exemplary case studies were developed that demonstrate several aspects of the g-Eclipse workbench, including the Workflow Builder's functionality. At a generic level the case studies followed a number of steps including: Creating a Grid project; Creating GRIA job descriptions as JSDLs; Constructing a workflow from the JSDL files; Translating the workflow into XScufl; Deploying the XScufl to the GRIA Job Service using the GRIA Workflow Deployer Application; Creating a Job Description (JSDL) for the newly deployed workflow application; Executing the new workflow JSDL; Visualizing the final output using the g-Eclipse visualization facilities.

One of the case studies carried out was a pharmaceutical application that implements several steps in the drug discovery process of analysing organic molecules. The Grid services used in this case study were derived from the SIMDAT project [21] that aimed to introduce Grid technology to various industry sectors, including the pharmaceutical industry. The workflow that was to be realized included three main sub-processes that were deployed on a GRIA site as individual applications:

- BLAST sequence analysis - The BLAST (Basic Local Alignment Search Tool) , described by Altschul in [15], is an application used to perform alignment anal-

ysis on biological sequences to discover similar structures in existing sequences databases.

- ANTIGENIC - This application analyzes a protein sequence to find potential antigenic regions.
- SRS3D [16] - As a final step, an application is used to transform the output of the sequence analyses into a data format that can be visualized by the g-Eclipse visualization plugins.

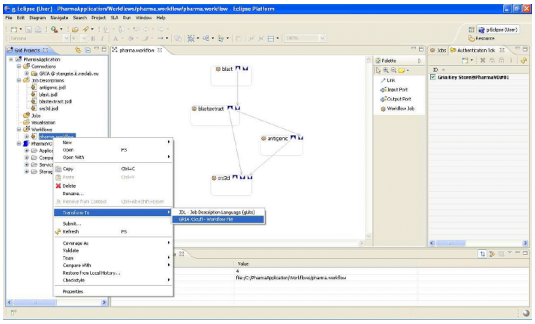

Figure 3. The pharmaceutical workflow being authored in the g-Eclipse Workflow Builder.

Each application step can take outputs directly from previous steps in the workflow, and it was possible to construct a workflow using the g-Eclipse Workflow Builder from these applications, as shown in figure 3. The workflow was simply the sequence of applications linked together in serial, each taking the output from the previous step. Figure 4 shows the output visualized in g-Eclipse from the constructed workflow.

Once the workflow had been constructed and deployed, users could then simply provide an intial input sequence, and as an output receive the analyzed and transformed data ready for visualization without any intervention in the intermediary steps as was before. Simplifying repetitive sets of processes such as the BLAST-ANTIGENIC-SRS3D case into a single application that hides the underlying workflow can greatly reduce the effort required in not just pharmaceutical analyses, but with any other scientific analysis processes that require repetitive and complex sub-activities.

6 Related work

There are two major scientific workflow design tools that are more mature and established than the g-Eclipse Workflow Builder. However it should be noted that g-Eclipse aims to be a generic Grid workbench, where the Workflow Builder is just one component of the workbench.

Taverna is a workflow composition and enactment tool which is a product of the myGrid UK e-Science project [17].

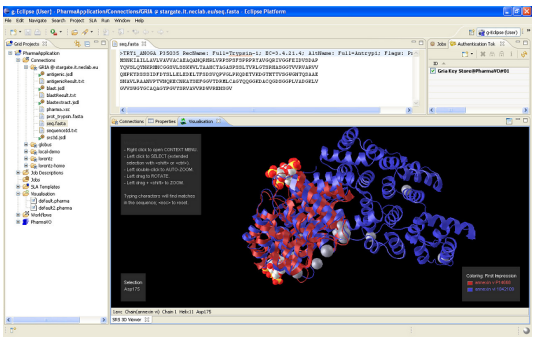

Figure 4. The output of the workflow displayed in the g-Eclipse Visualization Plugin.

Taverna aimed to enable bioinformatics analyses by allowing users to build workflows to access Web services that expose information repositories and compute resources. The Scufl workflow language (a precursor to XScufl) was developed by the project as there was no suitable standard for composing scientific workflows at the time. Like g-Eclipse, Taverna provides a graphical workbench for authoring workflows, however it also includes a workflow enactor called Freefluo. The g-Eclipse Workflow Builder does not provide any facility for workflow enactment and relies on the middleware implementations to deal with workflow execution. GRIA supports deployment of Taverna authored XScufl workflows via a GRIA plugin to the Taverna workbench. Taverna has been widely adopted by the UK eScience community and the details of this are discussed at length by Oinn et al in [18].

Kepler is a scientific workflow creation and execution tool that builds on the Ptolemy II concurrent component computation system [19]. Kepler allows users to build complex data-centric workflows in contrast to Taverna that is based on a singular-dataflow paradigm of workflows. As Kepler is builds on the Ptolemy system, it adopts Ptolemy's Modeling Markup Language (MoML) to persist workflows. Again, like g-Eclipse, Kepler provides a graphical workflow authoring tool, and like Taverna, also supports workflow execution. Kepler supports both a Web service and Grid resource workflow composition and has been adopted by a range of scientific fields such as biology, astrophysics and chemistry. The system is described in Altintas et al in [20].

7 Conclusions

In this paper we described the Workflow Builder tool that is part of the g-Eclipse Grid workbench. Authoring and deploying workflows is an essential part of developing scientific analyses techniques that require access to large data and

computing resources, and the g-Eclipse Workflow Builder goes some way to enabling users to author workflows and deploy them to different middlewares. Currently two Grid middlewares are supported by g-Eclipse, and the workbench framework is extensible enough to introduce support for other systems in the future. g-Eclipse is available as open source and can be downloaded from http://www.geclipse.eu or http://www.eclipse.org/geclipse

Acknowledgments

This work was supported in part by the EU under project (#FP6-2005-IST-034327), and the Eclipse Foundation. The authors would also like to thank the participating institutions of the g-Eclipse consortium and its project members.

References

[1] H. Kornmayer et al, "g-Eclipse Project," in *Proc. 2nd Austrian Grid Symp.*, Innsbruck, Austria, 2006.

[2] "g-Eclipse - Access the power of the Grid," 2009. [Online]. Available: http://www.geclipse.eu [Accessed: Oct. 14, 2009].

[3] I. Foster et al, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," in *Int. J. High Performance Computing Applicat.*, vol. 15, no. 3, pp. 200-222, Aug. 2001.

[4] Sciabá et al, Ed. (2009, Apr. 28), *gLite 3.1 User Guide (v1.2)* [Online]. Available: http://glite.web.cern.ch/glite/documentation/ [Accessed: Oct. 14, 2009].

[5] H. Gjermundrød et al, "g-Eclipse - An Integrated Framework to Access and Maintain Grid Resources," in *Proc. 9th IEEE/ACM Int. Conf. Grid Computing*, Tsukuba, Japan, 2008, pp. 57-64.

[6] D. Erwin, Ed. (2003) *UNICORE Plus Final Report* [Online]. Available: http://www.unicore.eu/documentation/files/erwin-2003-UPF.pdf [Accessed: Oct. 14, 2009].

[7] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," in *J. Comput. Sci. and Technology*, vol. 21, no. 4, pp. 513-520, Jul. 2006.

[8] M. Surridge et al, "Experiences with GRIA - Industrial Applications on a Web Services Grid," in *Proc. 1st Int. Conf. e-Science*, Melbourne, Australia, 2005, pp. 98-105.

[9] "GRIA - Service Oriented Collaborations for Industry and Commerce," 2009. [Online]. Available: http://www.gria.org [Accessed: Oct. 14, 2009].

[10] "Amazon Web Services," 2009. [Online]. Available: http://aws.amazon.com [Accessed: Oct. 14, 2009].

[11] A. Savva, Ed. (2005, Nov. 7) *Job Submission Description Language (JSDL) Specification, Version 1.0* [Online]. Available: http://www.ogf.org/documents/GFD.56.pdf [Accessed: Oct. 14, 2009].

[12] R. Allen, "Workflow: An Introduction," in *Workflow Handbook 2001*, L. Fischer, Ed. Lighthouse Point, FL: Future Strategies, 2000, pp. 15-38.

[13] Object Management Group (2003, May), *XML Metadata Interchange (XMI) Specification* [Online]. Available: http://www.omg.org/docs/formal/03-05-02.pdf [Accessed: Oct. 14, 2009].

[14] T. Oinn (2004, Apr. 7), *XScufl Language Reference* [Online]. Available: http://www.ebi.ac.uk/ tmo/mygrid/XScuflSpecification.html [Accessed: Oct. 14, 2009].

[15] S. Altschul, "Basic Local Alignment Search Tool," in *J. Molecular Biology*, vol. 215, no. 3. pp. 403-410, Oct. 1990.

[16] S.I. O'Donoghue et al, "The SRS 3D module: integrating structures, sequences and features," in *Bioinformatics*, vol. 20, no. 15, pp. 2476-2478, Apr. 2004.

[17] "The Taverna project website," 2009. [Online]. Available: http://taverna.sourceforge.net [Accessed: Oct. 14, 2009].

[18] T. Oinn et al, "Taverna: a tool for the composition and enactment of bioinformatics workflows," in *Bioinformatics*, vol. 20, no. 17, pp. 3045-3054, Jun. 2004.

[19] "The Kepler Project," (2009) [Online]. Available: https://kepler-project.org [Accessed: Oct. 14, 2009].

[20] I. Altintas et al, "Kepler: An Extensible System for Design and Execution of Scientific Workflows," in *Proc. 16th Conf. Scientific and Statistical Database Manage.*, Santorini, Greece, 2004, pp. 423-424.

[21] "SIMDAT Grids for Industrial Product Development," 2009. [Online]. Available: http://www.simdat.eu [Accessed: Oct. 14, 2009].