A Generic Grid Interface and Execution Framework for Biomedical Applications

Konstantinos I. Vegoudakis, Vassilis Koutkias, *Member, IEEE*, Andigoni Malousi, *Student Member, IEEE*, Ioanna Chouvarda, *Member, IEEE*, and Nicos Maglaveras, *Senior Member, IEEE*

***Abstract*—Nowadays, there is a growing demand for high computing power and large storage systems in the biomedical domain. Grid computing has recently gained a great deal of interest as an enabling technology towards realization of the e-Science vision, offering a highly flexible and controlled resource-sharing and collaborative environment. However, there is still a lack of user-friendly means to either access the Grid or enable the execution of existing biomedical software into Grid infrastructures. In this work, a generic interface enabling straightforward execution of biomedical applications to Grid infrastructures is presented, with respect to their input/output, software parameterization and attributes for compilation and execution, potentially external files used, etc. This functionality is supported by the definition of a generic XML schema for application description and, accordingly, by the construction of a dynamic graphical user interface based on this schema. Additionally, it provides the means to guide the user towards effective interaction with the Grid, in the entire lifecycle of application execution. The potential and applicability of the proposed approach, as well as the benefits of Grid computing, are illustrated via two existing biomedical applications, namely, the hyperparameter optimization of a Gaussian Support Vector Machine (SVM) classifier applied on human splice site prediction and the simulation of 2D cardiac propagation with the Luo-Rudy I model.**

I. Introduction

The conceptualization of modern science within a multidisciplinary rationale demands for collaborative environments, integration mechanisms, and advanced computing resources. Grid is an emerging technology that is expected to address both computing and storage problems, within a highly flexible and controlled resource-sharing and collaborative environment [1]. As great investments worldwide have been made the previous years towards the construction of Grid infrastructures, the Grid community expects to assess the actual potential of using the Grid [2]. However, up to now, the execution of existing applications in the Grid is a really complex procedure, primarily due to the lack of user-friendly tools and interfaces that may hide the intrinsic complexities of using the underlying technology.

Manuscript received July, 4, 2008. This work was supported in part by the Program of Postgraduate Studies in Medical Informatics, Aristotle University of Thessaloniki, Greece.

The authors are with the Lab of Medical Informatics, Medical School, Aristotle University of Thessaloniki, 54124 Thessaloniki, P.O. BOX 323, Greece (phone: +30 2310 999281; fax: +30 2310 999263; e-mail: kostasve@med.auth.gr, bikout@med.auth.gr, andigoni@med.auth.gr, ioanna@med.auth.gr, nicmag@med.auth.gr).

Among several application domains, Grid computing is envisioned to have a strong impact in the biomedical field, which is typically characterized by computationally intensive and complex applications [3-4]. In particular, the potential of Grid computing in healthcare has been recently elaborated in the context of the HealthGrid initiative [5], according to which the prospects of this technology endeavor are foreseen in e.g., evidence-based medicine, computational models of systems/organs, surgery simulation, genomic medicine, bioinformatics, pharmaceutical research, and large-scale epidemiological studies, to name a few.

In this regard, focusing on the realization of biomedical test-beds for the Grid, several projects have been established. For example, myGrid aims to exploit Grid technology, with an emphasis on the Information Grid, and provide middleware layers that fulfill the requirements of bioinformatics applications [6]. myGrid relies on the service-oriented paradigm, building high level services for data and application integration, such as resource discovery, workflow enactment, and distributed query processing. In the context of the EGEE (Enabling Grids for E-sciencE) project [7], which elaborates on the construction of the European Grid infrastructure, BIOMED Virtual Organization (VO) is worth mentioning that hosts biomedical applications in the underlying Grid infrastructure.

As far as user-friendly access to the Grid environment is concerned, in [8] an XML-based language is proposed that is designed to describe applications orchestrating Grid jobs, illustrated via a typical application example. The execution environment is also presented, supporting the proposed language, while it is shown how even simple Grid applications can be expressed in this language and benefit of the advanced capabilities of the execution environment. In addition, GuiGen is a comprehensive set of tools for creating customized graphical user interfaces (GUIs) [9], originally designed for the use in computational Grids. Via GuiGen, the user specifies the job (service) he/she wants to be executed and the Grid middleware autonomously decides on which system it is going to be executed.

Still, although the potential of Grid computing is well promising, there is limited number of applications that run on Grids, primarily due to the lack of simple and easy to use interfaces/tools that may be utilized by users with no strong technical background. Typically, users have to interact with Grid infrastructures by using Unix-like scripts that are

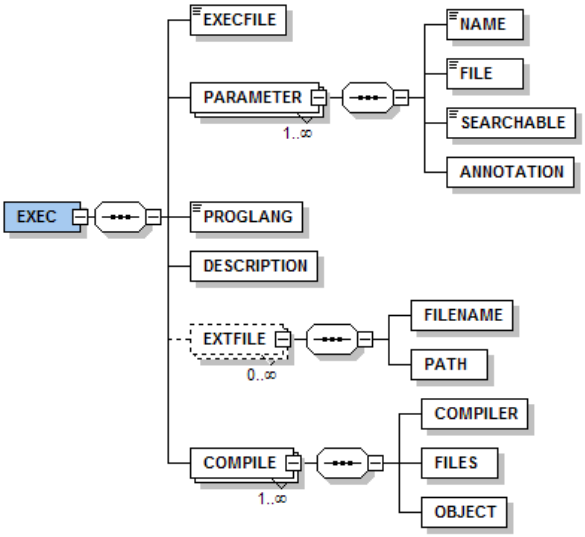

Fig. 1. Generic XML schema for application description.

offered by the underlying middleware, such as gLite WMS (Workload Management System), making execution of existing applications really complex. The required training on Grid computing and middleware, which seems mandatory even for users with rather strong technical skills, introduces additional barriers for potential new users.

In this work, a generic interface enabling straightforward execution of biomedical applications to Grid infrastructures is presented, which was developed based on XML (eXtensible Markup Language) and Java technologies. The proposed approach provides primarily a formal mean for applications description with respect to their input/output, software parameterization and attributes for compilation and execution, potentially external files used, etc., following a generic XML schema and, accordingly, dynamic user interface construction based on this schema. Equally important, it provides the means to guide the user of the application for effective interaction with the Grid, in the sense of job submission, data upload/download, etc. The potential and applicability of the proposed approach, as well as the benefits of Grid computing, are illustrated via two existing biomedical applications of bioinformatics and biomedical modeling context, respectively: a) the hyperparameter optimization of a Gaussian Support Vector Machine (SVM) classifier for human splice site prediction and b) the simulation of 2D cardiac propagation with the Luo-Rudy I model.

It has to be noted that the focus of this work is not on the design and development of Grid-optimized applications, in the sense of parallelization, but rather on enabling the execution of existing time-consuming and computationally intensive applications, especially ones requiring multiple executions in the context of parameter studies, as often met in the biomedical domain. Our approach envisions an environment enabling management of the entire lifecycle application execution, in a transparent way for the end-user.

The paper is structured as follows. First, the methodology adopted is presented, providing also relevant implementation details. Then, the test case biomedical applications are presented, illustrating the applicability of the proposed approach, as well as the results obtained and the benefits from their execution in a Grid environment. Finally, our future research directions and the conclusions drawn are discussed.

II. Methodology

A. Formal Application Description

Aiming to formally describe existing biomedical applications so that an appropriate GUI for their execution in a Grid environment is generated dynamically, a generic XML schema was defined. The proposed XML schema, though simple, encapsulates all the necessary requirements for an application to be compiled and executed on Grid, i.e. input/output, software parameterization, potentially external files used, and so forth. According to this schema, the XML document corresponding to the candidate Grid application is parsed, resulting in the dynamic generation of an appropriate interface for job submission and management. Both the corresponding script and a suitable JDL (Job Description Language) file for job definition are created automatically. The only thing the user has to do is to specify the execution details of the application of interest, according to the XML schema. Following this schema, any application for which a compliant XML document is provided may be executed.

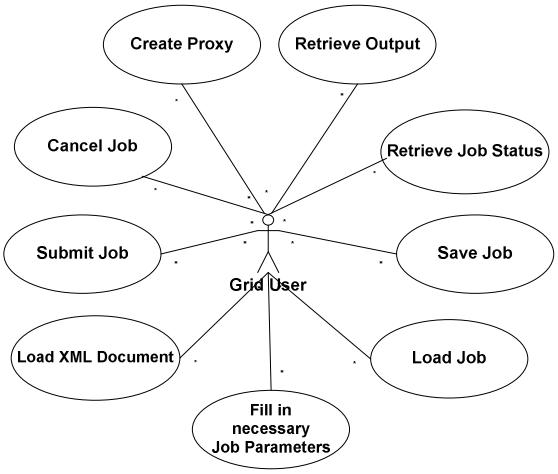

Fig. 2. Use cases diagram highlighting the user interaction patterns.

Figure 1 illustrates the major elements of this schema. In more detail, the *EXECFILE* element corresponds to the name of the executable file, while the parameters of the execution are defined in the *PARAMETER* element, with each application having one-to-many parameters. The *PROGLANG* element defines the programming language in which the application is written. Any external files such as libraries, dataset files etc., in which the application depends on are defined in the *EXTFILES* element. The *COMPILE* element is mandatory, in order to define the necessary

compilation features of the application, which takes place on the Worker Node (WN) that is expected to run.

B. User Interaction Patterns

The abovementioned XML schema constitutes the basis for the dynamic construction of an appropriate GUI allowing application execution in the Grid. The GUI encapsulates in a systematic way all the necessary steps one has to follow for job submission and management in the Grid. Figure 2 depicts the user interaction patterns that have been elaborated in terms of a use cases diagram. In more detail, a UML sequence diagram is depicted in Fig. 3, as regards the use cases Submit and Save Job. Specifically, in order to submit a job, first, the user has to load the XML document describing his/her application and, after filling in the appropriate application parameters, he/she can create a VOMS (Virtual Organization Membership Service) proxy certificate and submit a job. If the job is submitted successfully, the user is enabled to save the job in a file, referred by the identity returned from WMS, and load it whenever he/she wants upon request. In case of a successful job execution, the user is enabled to retrieve the results of the application.

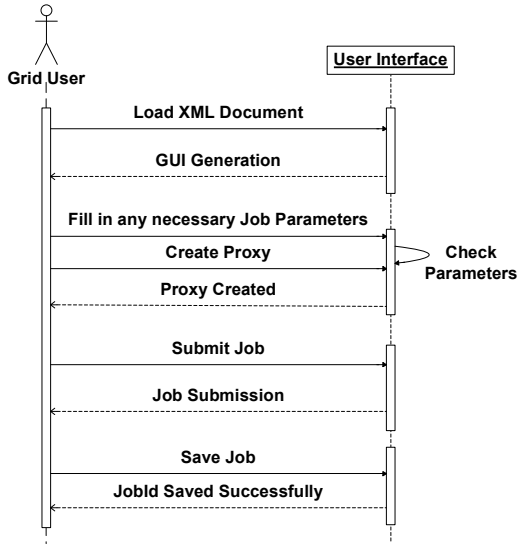

Fig. 3. Sequence diagram for the use case Submit and Save Job.

C. Dynamic User Interface Construction

In order to dynamically construct the GUI with respect to a valid XML application document, a software package was implemented. From a software engineering viewpoint, the package is depicted in Fig. 4 via a UML class diagram, presenting in a high level its major components (classes) and the relationships among them. In this design, the primary concern was flexibility. The main class, namely, *XMLInterFace* depends on the *DomParser* class, which offers the mechanism to parse the XML document. Classes *WMProxyJobSubmit*, *WMProxyJobCancel*, as well as *WMProxyGetOutput* (for job submission, job cancel and output retrieval, respectively) are independent of the

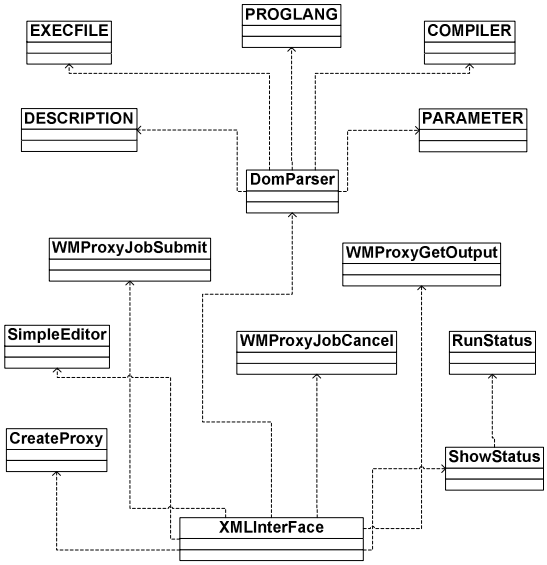

Fig. 4. UML diagram illustrating the main classes of the XML Interface.

DomParser class, enabling their code reuse for creating other interfaces.

D. Application Description – Implementation Details

The GUI constructed aims also to guide the user towards effective interaction with the Grid, with respect to job management in the entire execution lifecycle. For this reason, a control panel was created containing all the available user interactions patterns corresponding to the use cases defined. Figure 5 illustrates a screenshot of the GUI. The Utilities menu contains the XML Interface and the Grid FTP. The latter is used for file upload, when the application requires external files for execution, which should be stored on a Storage Element (SE). After loading a valid XML document, a form is generated dynamically containing all the control elements that the XML document defines, i.e. text boxes, buttons, etc., via which the user is enabled to submit jobs with different parameters. The Status button provides information for the state of the jobs on demand.

The GUI was developed in Java, using the WMProxy API (Application Programming Interface) to communicate with the Grid infrastructure (job submission, cancel, and output retrieval) [10]. As a Grid middleware, gLite 3.0 [11] was configured and installed in Scientific Linux Cern 3.0.8.

III. TEST CASES

In this section, two test cases involving existing biomedical applications are presented that were executed in the Grid following the proposed approach, aiming to assess its applicability and virtue.

A. Hyperparameter Optimization of a Support Vector Machine (SVM) for Human Splice Site Prediction

SVMs represent a major advance in supervised machine learning algorithms used in a wide-range of classification

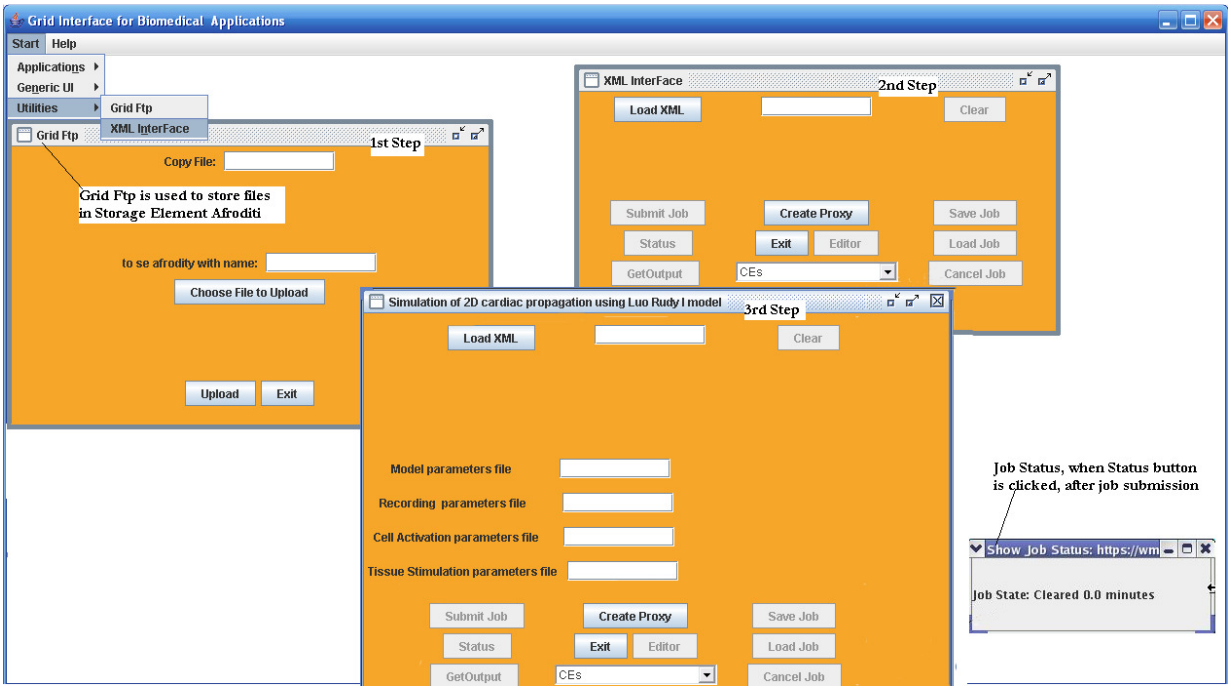

Fig. 5. Screenshot of the XML Interface illustrating the menu options, the step-by-step procedure of user interaction (the Grid FTP utility, the XML document load form and the control panel for jobs submission), as well as the job status information provided to the user on demand).

and regression problems [12]. The performance of a Gaussian SVM is calculated by setting the cost (C) of the classifier and the gamma (γ) parameter of the Gaussian kernel. These hyperparameters need to be optimized, in order to maximize the cross-validated accuracy of the classifier. An exhaustive and time-consuming search of the (C, γ) pair namely, grid search, is the most commonly used method for selecting the best parameter pair within a predefined log-scaled range of values.

In this example, an XML document according to the proposed schema was created to describe the 'grid search' application of the SVM classifier which is implemented using the libsvm (http://www.csie.ntu.edu.tw/~cjlin/libsvm/) library (Fig. 6). The goal is to identify the optimal hyperparameter pair so that the learning method can be optimally used for binary classification problems [13].

The GUI loads the corresponding XML document and automatically generates:

— A shell script of the following form:

```
grid.py [-log2c begin,end,step]
[-log2g begin,end,step] [-log2p begin,end,step]
[-v fold] [-svmtrain pathname] [-gnuplot pathname]
[-out pathname] [-png pathname]
[additional parameters for svm-train] dataset
```

Also, the appropriate information as regards compilation of files is included automatically in the shell script.

— An appropriate JDL file for the description of the job (resources, requirements, etc.).

— An interface with parameters and user dialogs for the proper submission/management of the job.

The constructed GUI enables the user to submit multiple jobs. Taking advantage of this option, grid search was implemented by splitting the (C, γ) parameters and searching into subsets for the optimal (C, γ) pair. Accordingly, the maximum predicted accuracy from the subsets was used to identify the (C, γ) pair giving the highest classification accuracy of all subsets.

The performance of the proposed Grid-based implementation of the optimal hyperparameter search was evaluated on a widely-known bioinformatics classification problem, namely, splice site prediction [14-16]. Probabilistic descriptions of the 102nt region centered at a candidate human splice site were used to build the feature space and the execution times of the hyperparameter optimization step were estimated on cross-validated training datasets of 1,000 and 10,000 positive and negative instances, respectively.

The results of this experiment are shown in Table I. The

```
- <EXEC>
    <EXECFILE>grid.py</EXECFILE>
  - <PARAMETER>
      <NAME>-log2c</NAME>
      <FILE>no</FILE>
      <SEARCHABLE>yes</SEARCHABLE>
      <ANNOTATION>cost parameter</ANNOTATION>
    </PARAMETER>
  - <PARAMETER>
      <NAME>dataset</NAME>
      <FILE>input</FILE>
      <SEARCHABLE>no</SEARCHABLE>
      <ANNOTATION>training dataset</ANNOTATION>
    </PARAMETER>
    <PROGLANG>python</PROGLANG>
  - <COMPILE>
      <COMPILER>g++</COMPILER>
      <FILES>svm.cpp svm-train.c</FILES>
      <OBJECT>svm-train</OBJECT>
    </COMPILE>
  - <EXTFILE>
      <FILENAME>svm-train.c</FILENAME>
      <PATH>../</PATH>
    </EXTFILE>
  </EXEC>
```

Fig. 6. Example XML document describing grid-search application.

tested log-scaled parameters for grid search were `log2c -6, 5` with step `1`, `log2g 1, 8` with step `1` and 9-fold cross-validation (`v 9`). Each line in Table I corresponds to submission of jobs for the same dataset and parameters, either using the split technique (last 4 rows) or not (first row). The results indicate a reduction in the execution time, which is able to reach 90.30%. At this point, it should be mentioned that the execution time cannot be reduced after a certain number of subsets, as the splits of the (C, γ) pair cannot be infinite.

TABLE I
EXECUTION TIMES FOR DIFFERENT NUMBER OF JOB SUBMISSIONS

Number of Jobs	Time (in hours)	% Reduction of Execution Time
1	100.45	Beginning of measurement
4 (2*2)	32.99	67.16%
6 (3*2)	24.90	75.21%
8 (4*2)	19.56	80.53%
16 (4*4)	9.75	90.30%

B. Simulation of 2D Cardiac Propagation using Luo-Rudy I Model

Simulation of 2D cardiac propagation using Luo-Rudy I model is an in-house application implemented in C++ language that simulates the 2D cardiac propagation under different conditions [17]. The 2D monodomain model of cardiac muscle is used. A rectangular sheet of cells is simulated with a grid of 40x200 elements, which corresponds to 40x50 cells. Each cell consists of 4 longitudinal elements (3 cytoplasmic and 1 junctional), with each element coupled by resistances. The Luo-Rudy I model is used for ionic kinetics. Regions of scar tissue can be defined, in order to introduce heterogeneity. During simulation, ionic and transmembrane currents, as well as action potential and gate parameters, are stored for certain predefined grid-points. This simulation application is useful in exploring the effect of different parameters variations on propagation characteristics. Therefore, testing the simulation with a series of parameter set combinations constitutes a valuable usage scenario.

The program runs with configuration files which define model-related parameters, recording sites, cellular activation, and extracellular stimulation parameters. These files are uploaded to a SE with the help of the Grid FTP utility. An XML document is constructed for this application as well, describing all the necessary requirements and resources for its execution. Upon loading the XML document, the GUI builds automatically a Unix shell script of the form:

```
#!/bin/sh
start_time=$(date +%s)
c++ matrices.cpp ion2kin.cpp LR_Model.cpp -o
model.o
./model.o LR_model.txt BR_positions.txt
LR_cell_9_7.txt LR_time_stim115.txt
finish_time=$(date +%s)
echo Time duration: $((finish_time - start_time))
secs. > time.txt
```

The script enables measurement of execution time of the simulation via the `start_time` and `finish_time` variables. This execution time can also be seen using the Status button of the GUI on demand.

Figure 5 depicts the job submission form that is dynamically created for this particular application. Using this form, the execution time was obtained from a randomly selected CE (Computing Element) for one simulation with four parameters. Table II compares the execution times between a CE and a typical desktop computer. Evidently, Grid prevails as it is faster than the desktop computer.

TABLE II
COMPARISON OF EXECUTION TIMES FOR THE SIMULATION OF 2D CARDIAC PROPAGATION USING LUO-RUDY I MODEL

Grid/Desktop Computer	Execution times in hours
Random CE of the Grid	7.84
Processor AMD Sempron 2600+	10.22

Besides any advantage as regards the execution time, a major benefit of such an approach is that a large number of parameter studies can be carried out. This feature enables the submission of huge number of jobs to the Grid with different parameter files (Table III), which can be executed in parallel. In this regard, scientists have the advantage to perform a large number of experiments and receive their results in a reasonable amount of time. This is clearly shown in Table III, depicting the results of four jobs submitted with different simulation parameters, four times each, as well as the average execution time calculated for each job. Another advantage of using the Grid is that a single desktop computer consumes a lot of computing power to execute multiple jobs, resulting in executing such kind of applications in a rather dedicated way.

TABLE III
COMPARISON OF EXECUTION TIMES FOR THE SIMULATION OF 2D CARDIAC PROPAGATION USING LUO-RUDY I MODEL (MULTIPLE JOB SUBMISSION)

Submission	Job1	Job2	Job3	Job4
1	19,150 sec	18,970 sec	19,165 sec	19,056 sec
2	19,134 sec	18,979 sec	29,658 sec	19,028 sec
3	19,142 sec	14,867 sec	19,156 sec	19,236 sec
4	19,139 sec	18,969 sec	19,159 sec	19,022 sec
Average time	5.32 hours	4.99 hours	6.05 hours	5.3 hours

IV. DISCUSSION - CONCLUSIONS

Grid computing has recently emerged as the main technical endeavor towards the realization of the e-Science vision [18], which aims to achieve global collaboration in key areas of science via the construction of the underlying

infrastructure that will enable it. In the last decade, several projects have been launched worldwide for the development of the appropriate components for the establishment of Grid infrastructures with respect to middleware technologies, focusing also on the exploitation and usage of these infrastructures with test-bed applications in various application domains. Especially for life sciences, the potential of Grid computing is rather strong, due to the complexity of the domain and the persistent computational and storage requirements.

Although major efforts have been made towards the abovementioned challenges, the existing Grid infrastructure lacks the means for user-friendly access towards its wide penetration and usage. In this work, we presented a generic Grid interface and execution framework for existing biomedical applications. The proposed approach relies primarily on a formal mean for applications description, in terms of their input/output, software parameterization and attributes for compilation and execution, potentially external files used, etc., according to a generic XML schema, enabling dynamic GUI construction based on this description. Equally important, it provides the means to guide the user for effective interaction with the Grid, with respect to job submission and management in its entire lifecycle.

The potential and applicability of the proposed approach, as well as the benefits of Grid computing, were illustrated via two test case biomedical applications, highlighting the reduction of execution times and the advantages of multiple job submission in parallel, a feature particularly suited for parameter studies which are rather common in biomedicine. The proposed approach enables the execution of existing time-consuming and computationally intensive applications, without requiring for example software re-engineering for MPI (Message Passing Interface) Grid optimization, provided that a compliant to the proposed schema XML document for their description is available. Besides biomedical applications, our work may be applicable to other scientific domains, enabling user-friendly access and execution of existing applications to the Grid.

The current implementation requires the installation of the gLite middleware in the computer hosting the GUI. This limitation is expected to be addressed, for example by developing a server-client tier, relying on either RMI (Remote Method Invocation) communication or on socket programming, to enable method invocation for job submission, canceling and output retrieval. In this regard, security concerns have to be further investigated. From a functional viewpoint, a forthcoming goal is also the enrichment of the approach supporting the definition and execution of application pipelines, in terms of workflows [6]. Towards this aim, existing workflow description languages and approaches applied in Grid are investigated [19].

REFERENCES

[1] I. Foster and C. Kesselman, "Concepts and architecture," in *The Grid: Blueprint for a New Computing Infrastructure* (2nd Edition), ser. Grid Computing, I. Foster and C. Kesselman, Eds. Elsevier, 2004, ch. 4, pp. 37-63.

[2] C. Goble, "The Grid needs you! Enlist now," in *On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, ser. LNCS, R. Meersman et al. Eds. Berlin Heidelberg: Springer-Verlag, 2003, vol. 2888, pp. 589-600.

[3] J.M. Alonso, J.M. Ferrero, V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "Three-dimensional cardiac electrical activity simulation on cluster and grid platforms," in *High Performance Computing for Computational Science - VECPAR 2004*, ser. LNCS, M. Daydé et al. Eds. Berlin Heidelberg: Springer-Verlag, 2005, vol. 3402, pp. 219-232.

[4] S. Benkner, C. Schröder, M. Lucka, and O. Steinhauser, "Grid services for parallel molecular dynamics with NAMD and CHARMM," in *Computational Science and Its Applications - ICCSA 2008*, ser. LNCS, O. Gervasi et al. Eds. Berlin Heidelberg: Springer-Verlag, 2008, vol. 5072, pp. 1036-1051.

[5] V. Breton et al., "The Healthgrid white paper," in *From Grid to Healthgrid: Proceedings of Healthgrid 2005*, ser. Stud Health Technol Inform., T. Solomonides et al. Eds. IOS Press, 2005, vol. 112, pp. 249-321.

[6] R.D. Stevens, A.J. Robinson, and C.A. Goble, "myGrid: Personalised bioinformatics on the information grid," *Bioinformatics*, vol. 19 Suppl. 1, pp. i302–i304, 2003.

[7] F. Gagliardi, "The EGEE European grid infrastructure project," in *High Performance Computing for Computational Science*, ser. LNCS, M. Daydé et al. Eds. Berlin Heidelberg: Springer-Verlag, 2005, vol. 3402, pp. 194-203.

[8] S. Reynaud and F. Hernandez, "A XML-based description language and execution environment for orchestrating grid jobs," in *Proc. of the IEEE Int. Conf. on Services Computing*, 2005, vol. 2, pp. 192-199.

[9] A. Reinefeld, H. Stüben, F. Schintke, and G. Din, "GuiGen: A toolset for creating customized interfaces for grid user communities," *Future Generation Computer Systems*, vol. 18, no. 8, pp. 1075-1084, 2002.

[10] gLite WM Proxy API Documentation. Available: http://trinity.datamat.it/projects/EGEE/wiki/wiki.php?n=WMProxyAPIAPIDocumentation

[11] gLite documentation. Available: http://glite.web.cern.ch/glite/documentation/

[12] V. Vapnik, Statistical Learning Theory. Wiley, 1998.

[13] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Department of Computer Science, National Taiwan University, Taipei 106, Taiwan, 21 May 2008. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[14] S. Sonnenburg, G. Schweikert, P. Philips, J. Behr, and G. Rätsch, "Accurate splice site prediction using support vector machines," *BMC Bioinformatics*, vol. 8(Suppl 10):S7, 2007.

[15] M. Pertea, X. Lin, and S.L. Salzberg. "GeneSplicer: A new computational method for splice site prediction," *Nucleic Acids Res*, vol. 29, no. 5, pp. 1185-1190, 2001.

[16] S. Degroeve, Y. Saeys, B. de Baets, P. Rouzé, and Y. van de Peer, "SpliceMachine: Predicting splice sites from high-dimensional local context representations," *Bioinformatics*, vol. 21, no. 8, pp.1332-1338, 2005.

[17] N. Maglaveras and I. Chouvarda, "Methodologies to evaluate simulations of cardiac tissue abnormalities at a cellular level," in *Digital Human Modeling*, ser. LNCS, V.G. Duffy. Ed. Berlin Heidelberg: Springer-Verlag, 2007, vol. 4561, pp. 694-702.

[18] T. Hey and A.E. Trefethen, "The UK e-Science core programme and the grid," *Future Generation Computer Systems*, vol. 18, no. 8, pp. 1017-1031, 2002.

[19] F. Neubauer, A. Hoheisel, and J. Geiler, "Workflow-based Grid applications," *Future Generation Computer Systems*, vol. 22, no. 1-2, pp. 6-15, 2006.