PoS PROCEEDINGS OF SCIENCE

UNICORE EMI-Execution Service realization towards improved open standards

Shahbaz Memon[1]
Forschungszentrum Juelich GmbH, Germany
E-mail: m.memon@fz-juelich.de

Morris Riedel
Forschungszentrum Juelich GmbH, Germany
E-mail: m.riedel@fz-juelich.de

Björn Hagemeier
Forschungszentrum Juelich GmbH, Germany
E-mail: b.hagemeier@fz-juelich.de

Bernd Schuller
Forschungszentrum Juelich GmbH, Germany
E-mail: b.schuller@fz-juelich.de

Michele Carpene
CINECA, Italy
E-mail: m.carpen@cineca.it

The EMI project unites a set of production Grid middleware technologies providing scientific communities a secure access to distributed and heterogeneous, compute and data resources. Within the EMI compute area, job management and monitoring are considered to be the most significant areas of work. Based on earlier Open Grid Forum (OGF) Production Grid Infrastructure (PGI) activities the existing standards and their adoption in the domain of job management on distributed computing infrastructures have been reviewed. As a consequence, several advanced execution service concepts have been identified that influenced the EMI-ES specification. The goal of this paper is to present the concepts of the EMI-ES interface and its information model that is required to manage, monitor, and model activities in production Grids. In this paper, we will delineate the architectural details of EMI-ES, and one of its 'proof of concept' realizations in UNICORE. The feedback of these activities is already part of the standardization process in OGF, and this paper puts existing Grid standards in context by comparing them with the proposed specification.

EGI Community Forum 2012 / EMI Second Technical Conference,
Munich, Germany
26-30 March, 2012

[1] Speaker

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence. http://pos.sissa.it

1. Introduction

EMI products have the mandate to support diversified scientific communities with non-trivial job management requirements. Initially, Grid middlewares followed an approach with proprietary interfaces and information models. But this approach was not productive as scientific communities tend to access resources in multiple infrastructures more often. Over the years, standard bodies such as OGF produced a set of Grid job management standards like OGSA-BES [3], JSDL [4], and GLUE2 [7]. It has been revealed from our experiences that these standards work well in production, but that some basic standards can also be improved with advanced execution service concepts. Hence, several of these concepts can be improved to facilitate an efficient execution and management of Grid applications on distributed computing infrastructures today. But in order to suggest several improvements for open standards, EMI [5] followed an approach to cover a set of concepts to be supported by ARC [11], gLite [9], and UNICORE [1], thus resulting in the EMI-ES specification. It consists of job management and monitoring interfaces, state model, activity description schema, and resource and activity representations.

The activity services are divided into two packages: Activity-Factory and Activity-Management. Activity-Factory consists of functionalities to initiate activities, managing resource manager information, and manages the delegation process used for data staging during activity lifecycle. Activity-Manager exposes a set of interfaces to manage the life cycle of activities, manages activity information, and also facilitates a delegation process which is also provided by Activity-Factory. Section 2 comprehensively elaborates each package by exploring their interfaces. The aforementioned functionalities are adopted within UNICORE using Web services technology. In the UNICORE architecture, there are services (like OGSA-BES or UNICORE Atomic Services [13]) through which clients can seamlessly interact with the services supported by XNJS [7] based execution management system. The EMI-ES specification per se and the lessons learned from the proof of concept implementations in EMI (here UNICORE) bear the potential to support many scientific communities that take advantage of the EMI distribution.

The specification comes up with a suite of well-defined interfaces, activity state model, support of vector operations, and an integrated job description model. The design and architectural elements have been discussed by the representatives of three major middleware technologies (ARC, gLite, UNICORE) within EMI. By having a support of one of the major middleware providers in a European Grid community, we see EMI-ES as a ‘proof-of-concept' specification for further standardization activities. EMI-ES succinctly expresses the implementation and operational experiences reflected by the production middleware technologies that are serving the diversified requirements of HPC and HTC based infrastructures. Consequently, we foresee its impact in fostering interoperability and integration required by the scientific communities engaged in executing complex scientific use cases. By having the implementations in EMI, we could practically implement a set of desired improvements aiming to improve the efficiency of production Grid applications. Pragmatically it realizes the set of concepts which could be improved in the existing standards space. With this effort we see EMI-ES becoming a potential source for the next generation of existing open Grid job management and description standards such as OGSA-BES, JSDL, and GLUE2, thus it will be a major contribution to the overall Grid standards community.

The remaining paper is structured as follows. Section 2 comprehensively describes the structure, main interfaces, and information model of the EMI-E**S**. Section 3 describes how UNICORE implementation has been adopted to this specification by highlighting its architecture layers from client tier to back end resource management system. Section 4 compares EMI-ES with the relevant OGF standards. Section 5 concludes the paper.

2. EMI-Execution Service (EMI-ES)

The EMI-ES is an initiative of the EMI project unifying ARC, gLite, and UNICORE to explore the possibility on a set of common job management and monitoring interfaces. EMI-ES targets the computing elements - the service entities representing capabilities of the resource management systems that are responsible for job execution and monitoring. The above mentioned EMI community members proposed requirements to cater shortcomings in the open standard interfaces implementation deployed in production infrastructures.

As shown in Figure 1, the specification defines web services interfaces grouped in two major components: *Activity-Factory* and *Activity-Manager*. The logical grouping is made to distinguish functionality of managing individual activities information and operations , and to represent a resource management abstraction - an entity allocates compute and storage resources to these activities.

Activity-Factory is responsible of initiating and creating activities, and giving a back-end resource agnostic resource management representation. The main interfaces building up this component are *Creation*, *ResourceInfo*, and *Delegation*.

Creation interface, as the name implies facilitate the creation of vector of activities upon client request. The client must provide the activity description in an ADL format; see later in this section for more details on ADL. This interface is designed in such a way that it can accommodate the data staging requirements wherein the client applications can push data to the job session directory, or server executing the job can pull the data from a source. Actually this situation reflects scenario requirement of the middleware developers of the EMI consortium.

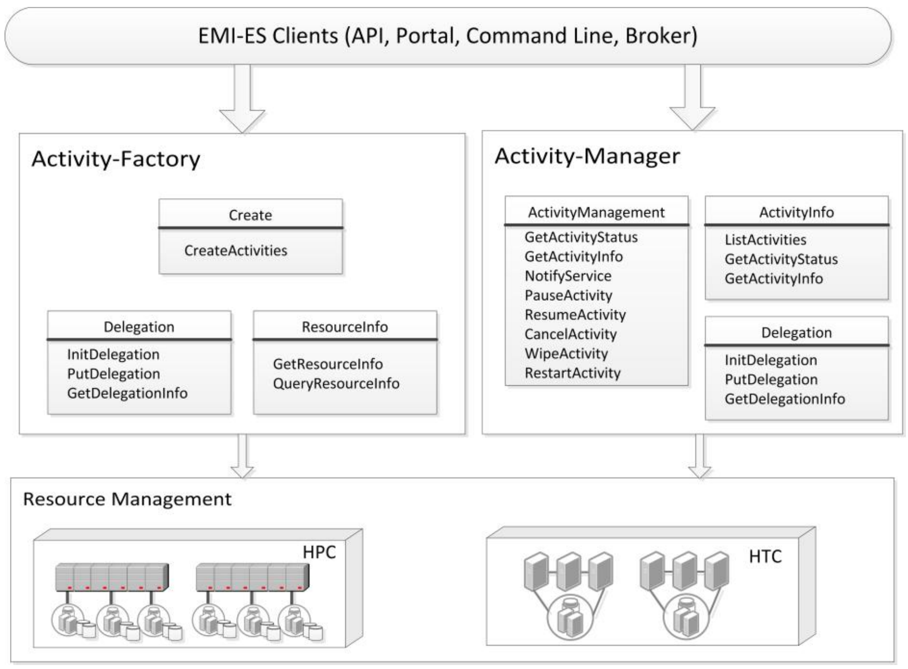

Figure 1: EMI-ES Architecture showing a client and a generic server side implementation with backend resource management abstraction

ResourceInfo interface provides the set of operations to project the local resource manager or batch system related information. The information retrieved by this entity must be compliant to the GLUE2 information model. Since the GLUE2 model is extensive in nature, thus it is essential to analyse GLUE2 entities against EMI requirements. Therefore, we identified ComputingManager and its sub concepts from GLUE2. Section 4 summarizes further the list of entities, their compositions and aggregates representing the resource manager model in the view of EMI-ES.

Delegation interface is an advancement in terms of managing the scenario where a job (before or after execution) needs to have a trust delegation while incorporating data- staging. In order to tackle this situation we proposed to associate this interface with Activity-Factory, which aims at facilitating an EMI-ES implementation to perform data-staging on behalf of the user sending jobs. This is an important use case in the landscape of Grid middleware operations, a common example where an incoming job requires stage-in/out data from/to “GridFTP” locations. While considering this use case as one of the core requirements of our stakeholders, it is mandatory for an implementation to use X.509 proxy certificates [20] as a basis for delegation mechanism. *Delegation* interface provide methods to initiate and issue delegation credentials on behalf of user. The delegation approach we employed here was originally defined by the GridSite proxy delegation mechanism [12].

Activity-Manager component shown in Figure 2 controls and monitors the individual set of activities and their state transitions. It is composed of three major interfaces: *ActivityManagement, ActivityInfo,* and *Delegation*. *ActivityManagement* interface encapsulates the functionality to monitor the set of activities statuses. It provides operations for a client to pause, resume, and cancel a vector of activities.

ActivityInfo port type is an interface allowing users to project the list of activities, their statuses, and the state information. For the sake of additional architectural flexibility and interface reusability the *Delegation* interface is also part of the *Activity-Manager* package.

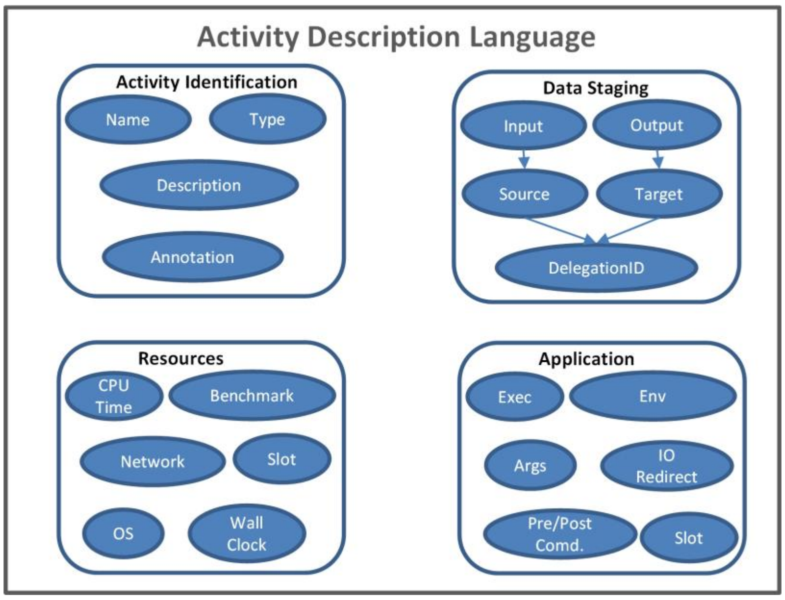

Figure 2: Activity Description Language (ADL)

Activity Description Language is part of EMI-ES specification and defines a data model to represent the job request used by clients. Figure 2 depicts the main entities and concepts of the ADL. It has four major elements: Activity Identification, Data Staging, Resources, and Application. Activity Identification and its sub concepts encapsulate the meta-information items needed to identify individual activities. Data staging primarily focuses on the specification of data sources and sinks in the job execution life cycle. This is also a model where delegation references are persisted. Resources element is an element through which service clients can specify the compute resources required to execute a job. Apart from the conventional compute resource elements such as OS, CPU, Memory, it also offers to specify runtime and parallel application environments. Application entity represents scientific application requirements by a client. Through this entity client can specify job details consisting of an executable location, list of arguments for an application, pre and post commands, and remote logging.

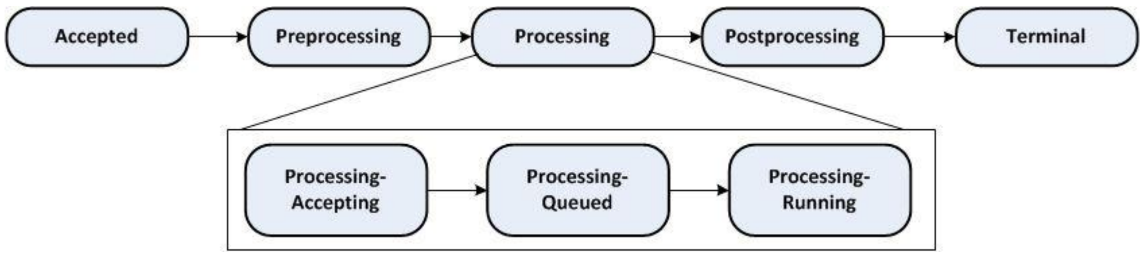

Figure 3: EMI-ES activity state model showing a usual job state transition

Activity state model of the EMI execution service provides the transition of an activity from inception to its completion. This state model is intended to reflect the status to the service clients. The implementations need not to have it internal, though they are free to use it if desired. Figure 3, describes the usual activity transition from *Accepted* to the *Terminal* status. These states are referred as the main activity states. In case an implementation needs to have sub states, the specification recommends using multiple attributes associated with each of the main states. For instance if an activity is in *Accepted* state, it might be possible that the service is still *Validating*, or deliberately halted by server as *Server-Paused*, or it is in the phase of *Provisioning* compute resources in a virtualised environment. The EMI Execution Service describes a rich set of attributes which could easily be applied against each of the main state.

3. EMI-ES in UNICORE Services Environment

UNICORE is an open standards based grid middleware providing a transparent, seamless, and secure access to the distributed high performance computing and storage resources. UNICORE supports a diversified platform agnostic deployment to any operating system. It exposes main stream compute and data management capabilities through SOAP based web services. It has been used in several European and national German infrastructure projects such as DEISA [2], EGI [10], and D-Grid [14]. From an application perspective, it is being used by various scientific communities, just to name a few are: Neurosciences, Virtual Physiological Human, and Biological sciences.

UNICORE is designed on the principles of service oriented, layered, and extensible architecture that ensures a seamless interface to the multiple resource management systems. Figure 4 depicts the client tier to interact with a remote job and data managed under the UNICORE service environment (UNICORE/X). The service environment is a layer which hosts SOAP based Web services. This layer exposes a backend functionality of the XNJS - an execution management system where the actual job incarnation takes place, and it knows how to communicate with a back end resource management system.

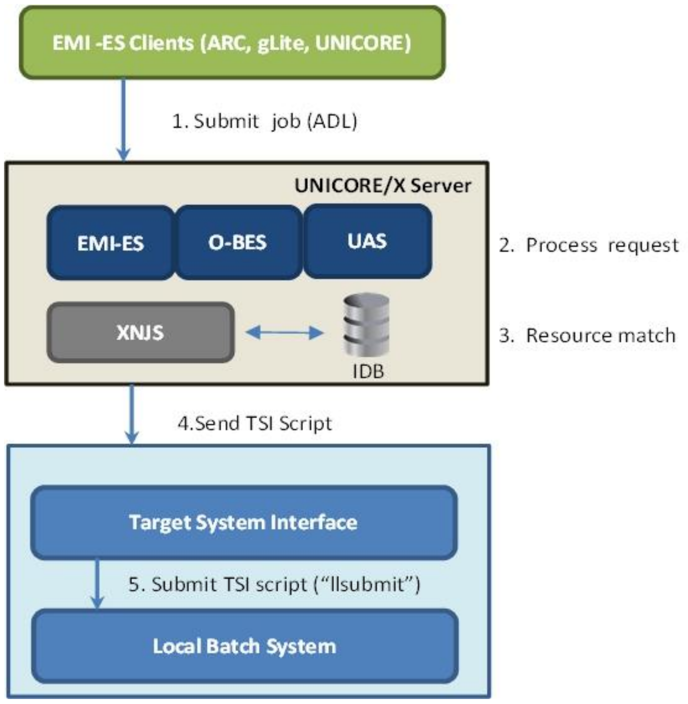

Figure 4: EMI-ES implementation in the UNICORE architecture

Figure 4 sketches a concrete realization of Figure 2, with more depiction of the server side implementation within the UNICORE services environment. EMI-ES is implemented as a Web service on the services layer which then connects the backend XNJS. From implementation perspective, XNJS is a dependency injection based framework that allows new components to be connected dynamically. By leveraging a potential of such a framework middleware developers can easily introduce new components without a need to rebuild the whole distribution. With this feature the implementation of EMI-ES is seen as yet another sub-component of XNJS. Specifically this tier implements the business logic to handle the semantics of ADL processing. Similarly XNJS realizes the JSDL processing. It implies that the UNICORE middleware could easily cope with the changes it requires in the future course of the specification if it is upgraded. Figure 4 also sketches the sequence of steps in which EMI-ES clients interacts with the UNCORE's EMI-ES implementation. Starting from the first step, client submits a job request in ADL form to the EMI-ES service. The service validates the request. Once it is validated, the XNJS takes care of the job submission by transforming the incoming request to the representation understood by the backend resource management system. While UNICORE considers the concern of various scientific communities accessing different kind of resources, it supports them by providing separate connectors for 13 resource management systems (including commercial and open source), just to name a few are, LSF [15], LoadLeveler [16], SLURM [17], TORQUE [18], etc.

4. EMI-ES and OGSA Job Management Standards

OGSA standards played a constructive role in bringing different Grid middleware communities to resolve interoperability and interoperation challenges together present in the current landscape of heterogeneous and distributed Grid infrastructures. This is related from several experiences that the scientific use cases often require different kind of compute resources which are geographically distributed should expose a common interface. Thus we see standards very useful in the provisioning of Grid resource access through a unified interface. Especially in the domain of Grid job management standards, which are believed to be one of the vital constituents of any middleware functionality. Among them OGSA-BES is a Web services based standard representing management and monitoring of computational jobs. Given the complexity of HPC and HTC paradigms, this interface is still being used as a common denominator in terms of job management and monitoring in the production research infrastructures. It is worth mentioning that, OGSA-BES despite of representing essential elements of managing Grid jobs, to some extent it is being criticized for not supporting contemporary computing and data requirements - from both the HTC and HPC perspectives. In order to deal with these concerns, EMI consortium and the Open Grid Forum, in the pursuit of standards adoption, collected more feedback in the form of use cases which actually were contemplated for further extensions to the existing job management and modelling standards such as OGSA-BES and JSDL. The main objective of this effort was to take a step further in tackling these requirements by not only supporting the well-known current physical computing infrastructures, but to address the use cases in which services are presumably deployed over Desktop Grids, virtual infrastructures, and clouds. Hence we see EMI-ES as a viable candidate to some of these requirements. After the requirements phase, EMI consortium together with an initial participation from the PGI (Production Grid Infrastructure) working group of OGF invested tremendous amount of efforts to formulate a specification with an aim to present a proof of concept supporting the next generation of existing OGSA-BES. While taking this initiative further, the major EMI representatives, such as, ARC, gLite, and UNICORE (as shown in the previous section) intending to demonstrate the cross-interface interoperability to the Grid standards community. Once the implementations and production experiences are concrete enough, EMI-ES will be contributed back with experiences to the OGF community as a building block for the next version of the OGSA-BES and JSDL standards.

To further support our argument, in Table 1, we holistically compared EMI-ES and OGSA-BES by drawing an intersection where both specifications meet with respect to scope and functionality. It also highlights the additional useful features offered by the EMI-ES specification. First column abstracts the area of job management and monitoring, the remaining columns differentiate both the interfaces in the given scope.

Scope	EMI-ES	OGSA-BES
Manage Activities	CreateActivities	CreateActivity (Single)
	Pause	--
	Resume	--
	Cancel	TerminateActivities (Vector)
Monitor Activities	Notify	Use of WSN
	Wipe	--
	ListActivities (return only activity identifiers)	--
	GetActivityStatus	GetActivityStatuses
	GetActivityInfo	GetActivityDocuments
Monitor Computing Service	GetResourceInfo	GetFactoryAttributesDocument
	QueryResourceInfo	Not in the spec (only WSRF [19] renderings)
Information Model	ResourceInfo (*Glue2:* ComputingManager, ComputingEndpoint, ExecutionEnvironment, ApplicationEnvironment)	FactoryAttributes (BES)
	ActivityInfo(*Glue2:*ComputeActivity)	ActivityDocument (JSDL)

Table 1: Comparison summary of EMI-ES and OGSA-BES

EMI-ES complements JSDL by proposing several advancements to model job description that appear in the form of ADL. ADL schema also introduces useful extensions to the GLUE2 resource information model, which are summarized under Table 2.

Application (ADL)	
Pre/Post-command	Commands should be executed before and after job execution
WipeTime	Timestamp until the terminated job is not removed
RemoteLogging	Points to the logging service where job logs are published
Notification	Information related to the notification service and job state change
Resource (ADL)	
Runtime Environment	When specified user doesn't need to provide executable path
Parallel Environment	Represent concept to model parallel jobs
ComputeActivity (Extensions to GLUE2 model)	
Stage-in/out directory	Separate elements to specify job stage-in and stage-out directory
Session directory	Activity execution directory
ComputingActivityHistory	Maintain Activity's provenance information
ComputingActivityProgress	Represents actual activity progress in percentages

Table 2: Overview of activity request and resource information model enhancements

5. Conclusion

In this contribution, we present the EMI-ES and its adoption in UNICORE. EMI-ES as a specification captures the Grid activity management and modelling requirements from the experiences of EMI's ARC, gLite, and UNICORE solutions. One of the 'proof of concept' implementations is in UNICORE, while ARC as well as gLite are also adopting it. Once all the EMI-ES adoptions within EMI are stable, we will take our endeavours to exercise interoperability with real application use cases providing insights on how future standard specifications can take production experience from this practical evaluation. By following this approach, we could anticipate its impact on overall scientific applications required to access resources in federated Grid infrastructures. We have also given the EMI-ES specification to the open standard working groups active in the area of job management and modelling (e.g. next generation OGSA-BES, JSDL, etc.) where the current focus is to take existing standards to the next versions of them.

6. Acknowledgements

This work has been partially sponsored by the European Commission funded EMI project (Grant Agreement INFSO-RI-261611).

References

[1] A. Streit and et al., *UNICORE 6 -Recent and Future Advancements*, Annals of Telecommunications, Next generation network and service management, Vol. 65, 11-12, pp. 757-762, Springer-Verlag, Berlin Heidelberg New York, ISBN 978-3-540-72226-7, 2010.

[2] W. Gentzsch and et al., *DEISA-Distributed European Infrastructure for Supercomputing Applications*, J. Grid Computing, Volume 9 Issue 2, June 2011 , 259-277

[3] I. Foster and et al., *OGSA Basic Execution Service Version 1.0*, GFD-R.108, 2008, available at http://www.ogf.org/documents/GFD.108.pdf, 15 April 2012

[4] A. Anjomshoaa and et al., *Job Submission Description Language (JSDL) Specification, Version 1.0*, GFD-R.136, 2008, available at http://www.ogf.org/documents/GFD.136.pdf, 15 April 2012

[5] B.Schuller, et al., *EMI Execution Service*, 21 Dec. 2011, Draft version 1.07, available at http://tinyurl.com/cn37ub4, 15 April 2012

[6] S. Andreozzi, et al., *GLUE Specification v. 2.0*, GFD-R-P.147, 2009, available at http://www.ogf.org/documents/GFD.147.pdf

[7] B. Schuller, R. Menday, A. Streit, *A Versatile Execution Management System for Next-Generation UNICORE Grids*, Euro-Par 2006: Parallel Processing, Lecture Notes in Computer Science, vol. 4375, pp. 195, 204, Springer Verlag, 2007

[8] *European Middleware Initiative*, http://www.eu-emi.eu, 15 April 2012

[9] E. Laure and et al. *Programming the Grid with gLite*, Computational Methods in Science and Technology, pp. 33-46, 2006.

[10] D. Kranzmüller, J. Marco de Lucas, and P. Öster, The European Grid Initiative (EGI): Towards Sustainable Grid Infrastructure. Springer US, pp. 61-66, 2010

[11] E. Edelmann, K. A. Happonen, J. Klem, J. Koivumki, T. Linden, and A. Pirinen, *Grid interoperation with ARC middleware for the CMS experiment* Journal of Physics: Conference Series vol. 219, no. 6, pp. 1-10, 2010

[12] M. Riedel, D. Mallmann, *Standardization Processes of the UNICORE Grid System*, Proceedings of 1st Austrian Grid Symposium 2005, pp. 191 - 203, Austrian Computer Society, ISBN 3-85404-210-2, 2006

[13] A. McNab and S. Kaushal, *The GridSite Proxy Delegation Service*, UK e-Science All Hands Conference, Nottingham, September 2006.

[14] Heike Neuroth, Martina Kerzel, Wolfgang Gentzsch, German Grid Initiative D -Grid. Niedersächsische Staats- und Universitätsbibliothek, 2007, ISBN 3938616997 (in German)

[15] *Platform LSF*, available at http://tinyurl.com/c9ln6b8, 23 April 2012

[16] *IBM Loadleveler*, available at http://www-03.ibm.com/systems/software/loadleveler/, 23 April 2012

[17] M. Jette and M. Grondona, *SLURM: Simple Linux Utility for Resource Management*, Linux Clusters: the HPC Revolution Conference, (San Jose, USA), 2003

[18] *TORQUE Resource Manager*, http://www.adaptivecomputing.com/products/open-source/torque/, 23 April 2012

[19] OASIS *Web Services Resource Framework (WSRF) – Primer v 1.2*, http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf, 23 April 2012

[20] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompsons, *Internet X. 509 Public Key Infrastructure (PKI) Proxy Certificate Profile*, RFC 3820, IETF, 2004