# Abstracting the Grid

Kaizar Amin[1,3] Mihael Hategan[1,2] Gregor von Laszewski[1,2,*] Nestor J. Zaluzec [1]

[*]Corresponding author: gregor@mcs.anl.gov

[1]Argonne National Laboratory 9700 S. Cass Avenue, Argonne IL 60439

[2]Computation Institute, The University of Chicago, 259 Ryerson Hall, Chicago, IL 60637

[3]Department of Computer Sciences, University of North Texas, P.O. Box 311366, Denton, TX 76203

Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2004), 11-13 February 2004 in A Coruña, Spain.

Also available as Argonne National Laboratory Preprint, ANL/MCS-P1101-1003

## Abstract

*The Grid approach allows collaborative pooling of distributed resources across multiple domains. However, the benefits of the Grid are limited to those offered by the commodity application development framework used. Several elegant and flexible application development frameworks support only specific Grid architectures, thereby not allowing the applications to exploit the full potential of the Grid. In order to initiate community interest to standardize a high-level abstraction layer for different Grid architectures, this paper introduces a collection of abstractions and data structures that collectively build a basis for an Open Grid Computing Environment.*

## 1. Introduction

The benefits offered by the Grid [1, 2] has resulted in the rapid expansion of the Grid community. An increasing number of advanced scientific and commercial applications have adopted the Grid architecture to realize their goals. Until recently, there were no standards in Grid development, a situation that resulted in several non-inter-operable Grid frameworks [3, 4, 5, 6].

The Global Grid Forum (GGF) [7] has presented itself as the much-required body that coordinates the process of standardization of the Grid development. The Open Grid Services Architecture (OGSA) [8] initiative of the GGF defines the artifacts for a standard service-oriented Grid framework based on Web services. OGSA enables different vendors to provide Grid service implementations in a variety of technologies, yet conforming to the GGF-defined OGSA standards, thus making them inter-operable.

The GGF Grid Computing Environment research group (gce-rg) [9] concentrates on the issues related to client-side Grid development tools. It investigates a collection of tools and user interfaces for accessing the services and functionality offered by the Grid. Since the gce-rg focuses on research aspects that provide the constructs for client development, service access tools, and the associated backend Grid services, it does not enforce any standards on the application development process. In order to address and fulfill the needs of the increasing Grid user community, we need a framework that provides a standard interface for client-side Grid development just as OGSA provides for server-side Grid development. Implementing such a framework will enable the Grid community to:

- develop client applications that will be inter-operable across multiple Grid backend implementations;
- provide re-usable code to support rapid prototyping of basic Grid access patterns;
- provide an open-source and extensible architecture that can be built collectively and incrementally based on community feedback; and
- access the same set of interfaces implemented in disparate technologies.

Motivated by the need to initiate an activity and foster community interest in the development of such a cross-architecture and technology-independent application development interface,

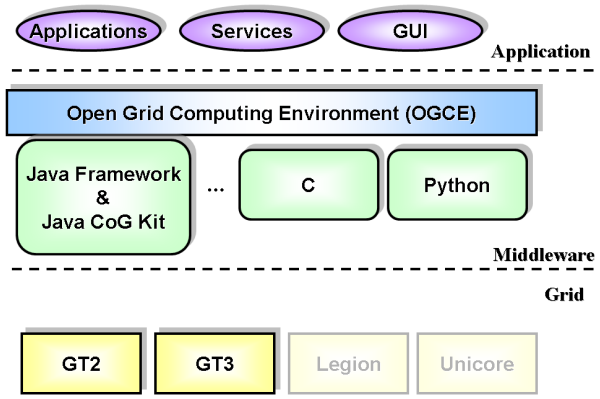


Figure 1: Development framework for Open Grid Computing Environment

we propose a set of initial services, data structures, interfaces, and the patterns of their interaction. We propose to call this standard client development environment the *Open Grid Computing Environment* (OGCE). The purpose of OGCE is to define and establish a set of standard Grid-centric patterns, services, and data structures that can be used by client developers as building blocks to develop sophisticated Grid-based applications. The data structures, interfaces, and their interactions can be implemented in different technologies as long as they conform to the common standards as proposed by the Commodity Grid Project (see Figure 1). Further, each of these implementations can provide transparent support to multiple Grid architectures and implementations.

The rest of this paper is organized as follows. Section 2 provides an overview of the related work being done in the domain of Grid application development frameworks. Section 3 describes a subset of the OGCE interfaces in detail. Section 4 discusses a reference implementation of the OGCE in Java. Section 5 gives a set of applications developed using the OGCE interfaces. Section 6 concludes our paper and outlines possible future research activities.

## 2. Related Work

A variety of application development toolkits are discussed in literature. In this section we describe a few of the most relevant projects and frameworks in relationship to our research.

The *jglobus* module of the Java CoG Kit [10, 11] (cog-jglobus) enables access to the Grid services provided by the Globus Toolkit. It offers low-level mappings to commonly used Grid services such as the Grid Security Infrastructure (GSI) [12], the Grid information services (MDS) [13, 14, 15], the Grid resource management services (GRAM) [16], and the Grid data access services (GridFTP) [17]. It also provides a range of low-level utility components that enhance the Grid functionality beyond those offered by the C implementation of the Globus Toolkit. The Java CoG Kit jglobus library is included as a part of the Open Grid Services Infrastructure (OGSI) [18] implementation distributed by Argonne National Laboratory. The Java CoG Kit enables a convenient development environment for Grid computing since 1997. It also distributes a large number of prototype plug-n-play graphical components such as the Grid desktop, Grid shell, Grid FTP manager [19], and the Grid workflow manager to enhance the functionality of Grid-aware applications [20].

The Grid application framework for Java (GAF4J) [21] provides an abstraction layer over the underlying Grid infrastructure. It offers a suite of abstraction classes that hides the low-level implementation from the end-users. The current implementation of GAF4J uses the Java CoG Kit jglobus library to provide its functionality over a Globus Toolkit-enabled Grid infrastructure. It includes a programming model that introduces a simple Task object as one of its main contributions. As the Globus Toolkit does not contain such an abstraction level, it provides a higher level of abstraction than does the Globus Toolkit.

The abstract job object (AJO) [4] provides the abstraction layer for the UNICORE protocol layer (UPL) in UNICORE Grids. It defines the basic communication abstraction between the client applications and the underlying Grid protocols. The Grid interoperability project (GRIP) creates a bridge between the UNICORE and the Globus Toolkit frameworks [22]. It translates UNICORE requests (in AJO and UPL) for remote job execution, information queries, authorized access into corresponding Globus Toolkit constructs. The GRIP framework allows a UNICORE client seamless access to both the Globus Toolkit and UNICORE Grids.

The literature about the EveryWare toolkit [23] reports that it enables Grid-aware applications to submit tasks to multiple Grid architectures. It offers the flexibility to submit jobs to the Globus Toolkit, Legion [5], Condor [6], and NetSolve [24] architectures. The toolkit provides a set of perfor-

mance forecasting services to optimize system performance by making real-time performance predictions and submitting the tasks to the appropriate resources. However, this project seems no longer active and due to the many recent cahnges in the Grid community will be outdated.

Each of the frameworks described in this section either concentrates on a specific technology or caters to the needs of a particular backend Grid implementation. This paper proposes a technology- and architecture-independent abstraction layer that provides true interoperability across multiple Grid implementations. Adaptation to other Grid architectures can be enabled while implementing the appropriate interfaces defined within this framework that are derived from the the most common use patterns in Grids.

## 3. OGCE

One of the most important usage patterns in Grid computing is the execution of a Grid task. An extension to this basic Grid execution pattern is a Grid workflow pattern that enable the user to submit a set of Grid tasks along with an execution dependency. Therefore, the initial design of OGCE concentrates on providing the artifacts required to support these important usage patterns. Other Grid patterns can be supported by extending the flexible OGCE design based on community feedback.

Although the Globus Toolkit and Condor focused on patterns related to file transfer and job execution, they did not sufficiently address the issues of general task patterns. In Condor, the focus is clearly on high throughput brokering by concentrating on a limited pattern for job execution with integrated file input and output. However, we believe that the separation of file transfer, information services, and job execution is necessary to allow general workflows that can also enable the integration of non-Grid related tasks. Such a high level abstraction layer has also the consequence that the architecture to be deployed may be simplified as documented in [25]. As an example for such a simplification we point out that there is a great deal of overlap by dealing reliably with information queries, file transfers, and job submissions. Hence, it warrants the introduction of our high level patterns that make the top down development of sophisticated services a reality. This contrasts and enhances the bottom up design conducted by many ongoing Grid research efforts.

Figure 2 shows the class diagram of the OGCE framework. A detailed listing of the attributes and functions for each class has been omitted for simplicity. In the rest of this section we describe the important entities designed and their semantics as a part of OGCE.

### 3.1. Task

A *Task* is the atomic unit of execution in OGCE. It represents a generic Grid functionality including remote job execution, file transfer request, or information query. It has a unique identity, a security context, a specification, and a service contact.

The task identity helps in uniquely representing the task across the Grid. The security context represents the abstract security credentials of the task. It is apparent that every underlying Grid implementation enforces its own security requirements therefore making it necessary to abstract a generalized security context. Hence, the security context in OGCE offers a common construct that can be extended by the different implementations of OGCE to satisfy the corresponding backend requirement. The specification represents the actual attributes or parameters required for the execution of the Grid-centric task. The generalized specification can be extended for common Grid tasks such as remote job execution, file transfer, and information query. The service contact associated with a task symbolizes the Grid resource required to execute it.

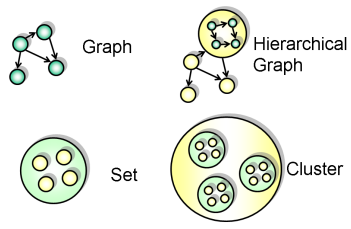


Figure 3: Use cases for CompositeTask and SuperTask

### 3.2. CompositeTask and SuperTask

A *CompositeTask* in OGCE provides a building block for expressing complex dependencies between tasks. All significantly advanced applications require mechanisms to execute client-side workflows that process the tasks based on user-defined dependencies. Hence, the data structure representing the

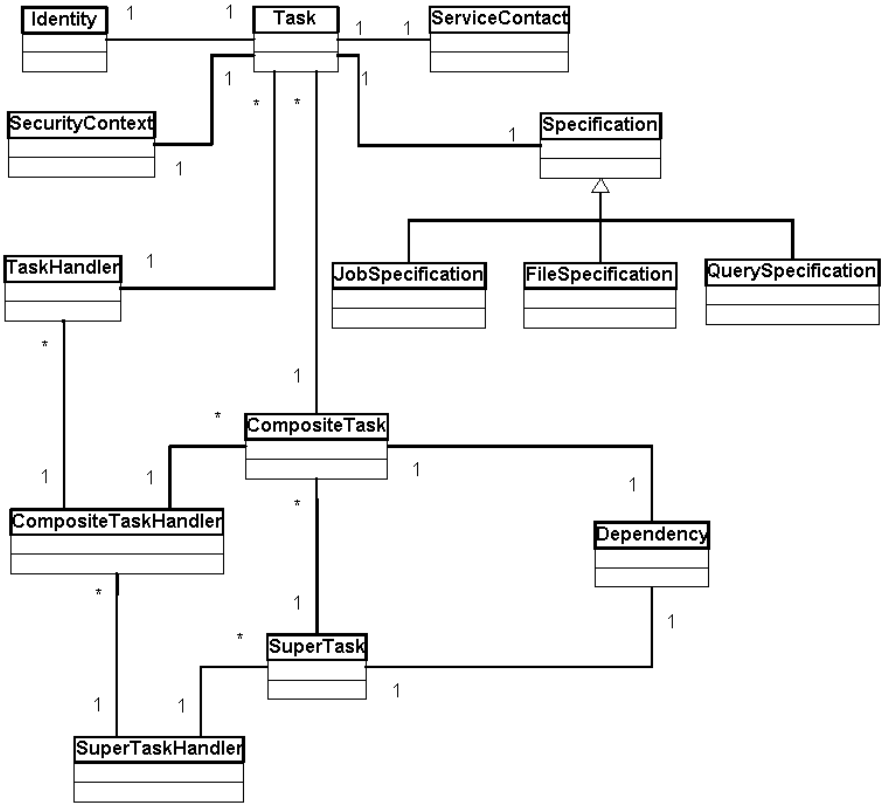


Figure 2: UML Class Diagram for OGCE

composite task aggregates a set of tasks and allows the user to define dependencies between these tasks. In graph theoretical terms, a composite task provides the artifacts to express workflows as a directed graph (see Figure 3). Hence, for a graph $G(V, E)$, the set of tasks represent the set of vertices $V$ and the dependencies correspond to the set of edges $E$. Apparently, the composite task can also represent a set of independent tasks with an empty set of edges.

A *SuperTask* in OGCE represents a workflow of composite tasks aggregating a set of composite tasks and associating it with a dependency. In graph theoretical terms, a super task represents a hierarchical graph, that is, a graph of graphs (see Figure 3). It can also symbolize a cluster of task sets with a null dependency. We note that with a combination of task, composite task, and super task a flexible framework is provided to express Grid workflows conveniently.

### 3.3. Handlers

OGCE contains the *TaskHandler*, the *CompositeTaskHandler*, and the *SuperTaskHandler* to process a task, a composite task, and a super task, respectively. The task handler provides a simple interface to handle a generic Grid task submitted to it. It is capable of categorizing the tasks and providing the appropriate functionality for it. For example, the task handler will handle a remote job execution task differently than a file transfer request task. OGCE does not impose any restrictions on the implementation of the task handler as long as its working is transparent to the end user. We note that this module is backend-specific and will have a separate implementation for each Grid architecture it supports.

The composite task handler and the super task handler provide a similar functionality as the task handler interface. However, they have an additional responsibility of enforcing the dependency on the graph-like task sets submitted to them. They can be implemented as advanced workflow engines coordinating the execution of tasks on corresponding Grid resources honoring the user-defined dependencies.

Listing 1: Sample code snippet demonstrating the level abstraction offered by the OGCE interfaces

```
private void prepareAndSubmitTask() {
  // Create a Grid task for job execution
  Task task =
      new TaskImpl("myGridTask",
              Task.JOB_SUBMISSION);

  // Set the provider for this task as GT2.
  // In order to submit this task as a GT3
  // task, simply set the provider to GT3
  // and change to serviceContact to point
  // the GT3 job execution service.

  task.setProvider("GT2");

  // create a remote job specification
  JobSpecification spec =
      new JobSpecificationImpl();

  // set all the job related parameters
  spec.setExecutable("/bin/workflow");
  spec.setStdInput("job-input");
  spec.setStdOutput("job-output");
  spec.setStdError("job-error");

  // bind the specification to the task
  task.setSpecification(spec);

  // create a security context
  SecurityContextImpl securityContext =
      new GlobusSecurityContextImpl();

  // selects the default credentials
  securityContext.setCredential(null);

  // bind the security context to the task
  task.setSecurityContext(securityContext);

  // create a service contact
  ServiceContact serviceContact =
      new ServiceContactImpl("server:port");

  // associate the contact with the task
  task.setServiceContact(serviceContact);

  //create a task handler
  TaskHandler handler =
      new GenericTaskHandlerImpl();

  // submit the task to the handler
  handler.submit(task);
}
```

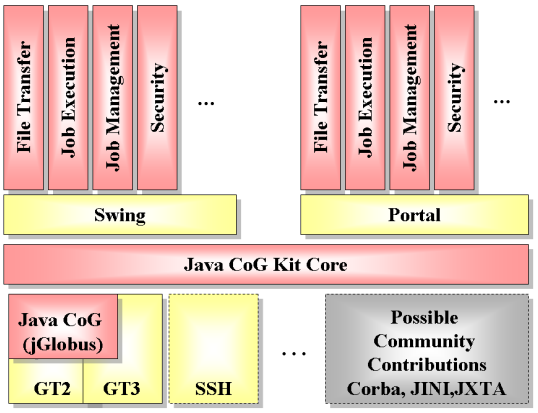


Figure 4: The Java CoG Kit (core module) provides a reference implementation of the OGCE interfaces simplifying the development of Grid computing environments using portals or applications.

## 4. Reference Implementation

To validate the OGCE design, we have developed a reference implementation for it in Java. Our reference implementation is available as the *core* module of the Java CoG Kit (cog-core). At time of writing of this paper the core module is in prototype stage. The cog-core uses cog-jglobus to provide support for Globus Toolkit v2 and implements an OGSA-compliant handler for supporting the Globus Toolkit v3. Further, cog-core also supports secure shell (ssh) based job execution and file transfer mechanisms. Although the ssh-based mechanisms are neither recommended nor suitable for large scale Grid production environments, they offer convenient facilities for quick prototyping of Grid applications in the absence of a production Grid. Once the required Grid infrastructure is in place, the application can seamlessly shift from an ssh-based mechanism to a Grid-based solution. Listing 1 shows a sample code snippet demonstrating the abstractions offered by the OGCE interfaces. In this example we create a task for submission to a Globus Toolkit v2 job submission service. It is important to note that with a switch of a single parameter, the provider, the same task could be submitted to a Globus Toolkit v3 execution service.

## 5. Application

Applications of OGCE are manifold. It can be used as convenient abstraction for elementary Grid

functionality to develop Grid middleware, high-level Grid services, and Grid applications (see Figure 4). As a consequence, several components distributed with the Java CoG Kit [10] have been rapidly prototyped using the cog-core reference implementation (see Figure 5). These components include a Grid workflow [26], Grid desktop, Grid shell, Grid job managers, and Grid portals [27]. These abstractions provided by the Java CoG Kit allow the application developer to concentrate on the development of higher level middleware components instead of keeping up with the changes that are currently underway in the Grid architecture.

### 5.1. Position-resolved Diffraction for Nanoscale Structures

To further test the usefulness of our implementation we are prototyping a Grid computing environment for the analysis of nanoscale structures, explained in more detail in [26]. As a part of this effort, a new experimental technique, named position-resolved diffraction, is being developed to study nanoscale structures at the Argonne National Laboratory's advanced analytical electron microscope [28]. With this technique, a focused electron probe is sequentially scanned across a two-dimensional field of view of a thin specimen. At each point on the specimen, a two-dimensional electron diffraction pattern is acquired and stored (see Figure 6). As much as one terabyte of data can be taken during such an experiment. This analysis of the data requires a resource rich Grid infrastructure to fulfill the real-time constraints. The results need to be archived, remote compute resources need to be reserved and made available during an experiment, and the data needs to be moved to the compute resources where they will be analyzed. Finally, they need to be presented in a meaningful way to the scientist. Our cog-core patterns provide a convenient initial abstraction for the application scientist to develop this sophisticated environment while hiding as much of the complexity of the underlaying Grid middleware.

The need for such a flexible infrastructure is demonstrated through a simple experiment flow depicted in Figure 7. The elementary logic of the instrument control can be expressed in a sequence of processes that depend on each other. They include the data acquisition, the data backup, the data analysis, and the result display.

By formulating the processes related to the experiment through a graphical interface the scientist will be able to conveniently interact through a graphical component with the instrument and experiment resources.

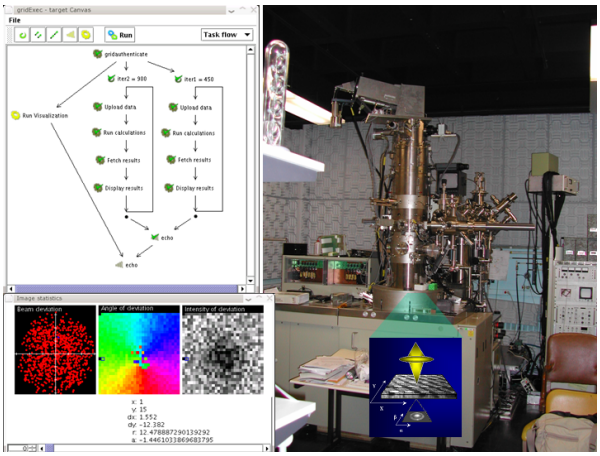


Figure 6: The data analysis for the electron microscope is formulated as workflow that uses Grid resources. The progress of the calculation is updated in real-time

## 6. Summary and Future Work

This paper is motivated by the need to develop a technology independent, architecture agnostic, open, and extensible "lingua franca" for Grid application development. We have presented our initial design of the OGCE abstraction interfaces. As a proof of concept we have developed a reference implementation of OGCE supporting the Globus Toolkit versions 2 and 3. We showed the convenience of the OGCE abstractions by using the reference implementation in a suite of target applications.

The implementations of OGCE can be used by clients to support multiple Grid architectures or by backend services to provide cross-platform bridging capabilities. The OGCE design is a work in progress and will undergo changes based on community feedback while at the same time monitoring the development of OGSA and OGSI. Ongoing research is concentrating on extending the OGCE framework for providing abstractions for scheduling and resource brokering mechanisms. Our reference implementation is available as the *core* module of the Java CoG Kit.

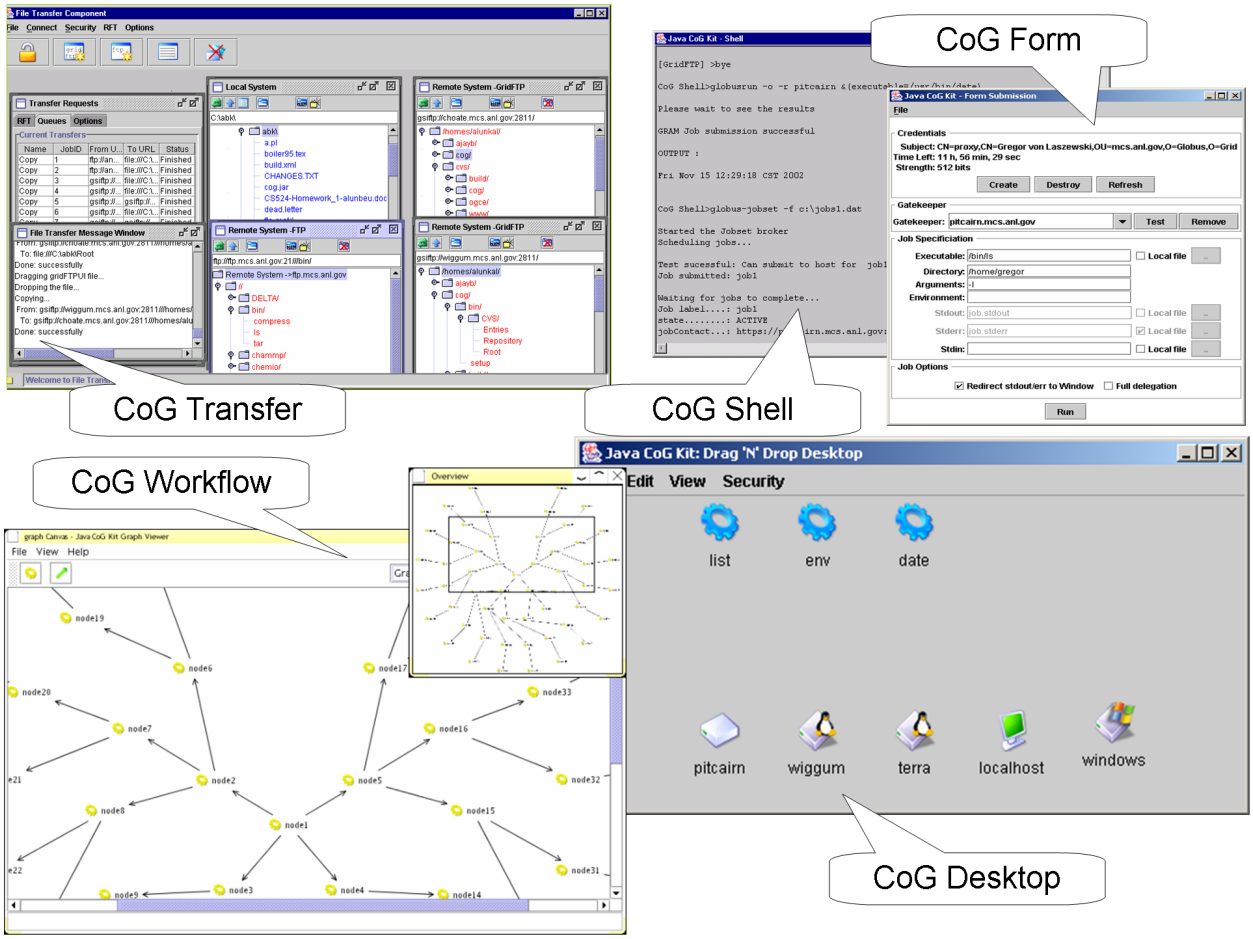


Figure 5: Development of portable high-level graphical interfaces is simplified by using OGCE interfaces.

## Acknowledgments

This work was supported by the Mathematical, Information, and Computational Science Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-Eng-38. DARPA, DOE, and NSF support Globus Project research and development. The Java CoG Kit Project is supported by DOE SciDAC and NSF Alliance.

## References

[1] G. von Laszewski and P. Wagstrom, "Gestalt of the Grid," in *Performance Evaluation and Characterization of Parallel and Distributed Computing Tools*, ser. Series on Parallel and Distributed Computing. Wiley, 2003, (to be published). [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--gestalt.pdf

[2] G. von Laszewski and K. Amin, *Grid Middleware*. Wiley, 2004, ch. Middleware for Commnications, to be published. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--grid-middleware.pdf

[3] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputing Applications*, vol. 15, no. 3, 2002. [Online]. Available: http://www.globus.org/research/papers/anatomy.pdf

[4] "Unicore," Web Page. [Online]. Available: http://www.unicore.de/

[5] A. S. Grimshaw and W. A. Wulf, "The Legion Vision of a Worldwide Virtual Computer," *Communications of the ACM*, vol. 40, no. 1, pp. 39–45, January 1997. [Online]. Available: http://legion.virginia.edu/copy-cacm.html

[6] D. Thain, T. Tannenbaum, and M. Linvy, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley, 2003, no. ISBN:0-470-85319-0, ch. Condor and the Grid.

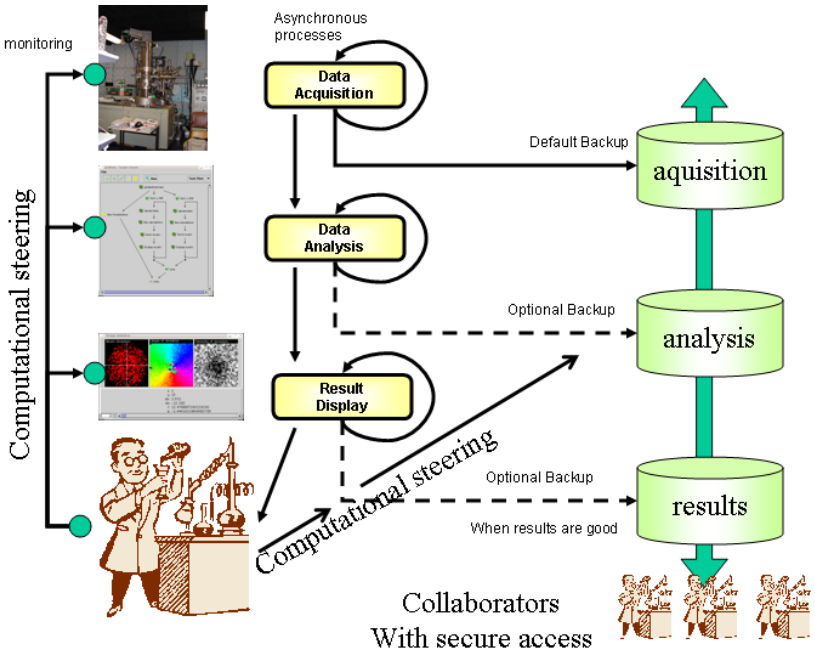


Figure 7: Asynchronous processes define a workflow that is steered by the scientist to support the problem solving process with the help of abstract Grid tasks


[7] "The Global Grid Forum Web Page," Web Page. [Online]. Available: http://www.gridforum.org

[8] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Web page, Jan. 2002. [Online]. Available: http://www.globus.org/research/papers/ogsa.pdf

[9] "Grid Computing Environment Research Group," Web Page. [Online]. Available: http://www.computingportals.org/

[10] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8-9, pp. 643–662, 2001. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--cog-cpe-final.pdf

[11] "Java CoG Kit," Web Page. [Online]. Available: http://www.globus.org/cog/

[12] "Grid Security Infrastructure," Web Page. [Online]. Available: http://www.globus.org/security/

[13] G. von Laszewski, S. Fitzgerald, I. Foster, C. Kesselman, W. Smith, and S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Computations," in *Proceedings of the 6th IEEE Symposium on High-Performance Distributed Computing*, 5-8 Aug. 1997, pp. 365–375. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/fitzgerald--hpdc97.pdf

[14] "The Monitoring and Discovery Service," Web Page. [Online]. Available: http://www.globus.org/mds

[15] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing." in *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing*. San Francisco, CA: IEEE Press, 7-9 Aug. 2001, pp. 181–184. [Online]. Available: http://www.globus.org/research/papers/MDS-HPDC.pdf

[16] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A Resource Management System for Metacomputing Systems," in *PPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, 1998, pp. 62–82.

[17] B. Allcock, L. Liming, and S. Tuecke, "GridFTP: A Data Transfer Protocol for the Grid." [Online]. Available: http://www.gridforum.org/Documents/Drafts/GF5%20Drafts/gridftp_intro_gf5.pdf

[18] "Open Grid Services Architecture (OGSA)," Web Page. [Online]. Available: http://www.globus.org/ogsa

[19] G. von Laszewski, B. Alunkal, J. Gawor, R. Madhuri, P. Plaszczak, and X.-H. Sun, "A File Transfer Component for Grids," in *Proceedings of the International Conferenece on Parallel and Distributed Processing Techniques and Applications*, H. Arabnia and Y. Mun, Eds., vol. 1. Las Vegas: CSREA Press, June 23-26 2003, pp. 24–30. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--gridftp.pdf

[20] G. von Laszewski, B. Alunkal, K. Amin, J. Gawor, M. Hategan, and S. Nijsure, "The Java CoG Kit User Manual," Argonne National Laboratory, Mathematics and Computer Science Division, 9700 S. Cass Ave, Argonne, IL 60439, U.S.A., MCS Technical Memorandum ANL/MCS-TM-259, Mar. 14 2003.

[21] A. Jhoney, M. Kuchhal, and Venkatakrishnan, "Grid Application Framework for Java (GAF4J)," IBM Software Labs, India, Tech. Rep., 2003. [Online]. Available: https://secure.alphaworks.ibm.com/tech/GAF4J

[22] D. Snelling, S. van den Berghe, G. von Laszewski, P. Wieder, J. MacLaren, J. Brooke, D. Nicole, and H.-C. Hoppe., "A unicore globus interoperability layer," Web page, Nov. 2001. [Online]. Available: http://www.grid-interoperability.org/D4.1b_draft.pdf

[23] R. Wolski, J. Brevik, G. Obertelli, N. Sprong, and A. Su, "Writing Programs that Run EveryWare on the Computational Grid," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1066–1080, October 2001.

[24] H. Casanova and J. Dongarra, "NetSolve: A Network Server for Solving Computational Science Problems," *International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 3, pp. 212–223, 1997.

[25] G. von Laszewski, J. Gawor, C. J. Peña, and I. Foster, "InfoGram: A Peer-to-Peer Information and Job Submission Service," in *Proceedings of the 11th Symposium on High Performance Distributed Computing*, Edinbrough, U.K., 24-26 July 2002, pp. 333–342. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--infogram.ps

[26] K. Amin, M. Hategan, G. von Laszewski, A. Rossi, S. Hampton, and N. J. Zaluzec, "GridAnt: A Client-Controllable Grid Workflow System," in *37th Hawai'i International Conference on System Science*, Island of Hawaii, Big Island, 5-8 Jan. 2004. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--gridant-hics.pdf

[27] G. von Laszewski, N. Maltsev, D. Sulakhe, S. Nijsure, S. Shankar, M. Hategan, E. Marland, A. Rodrigez, K. Amin, B. Alunkal, V. Nefedova, and G. X. Yu, "Developing Scientific Portals through Portal Middleware (with application to Gnare, a portal to Genome Analysis Research Environment)," in *Midwest Software Engineering Conference*. Chicago: DePaul University, June 5th 2003, p. 194.

[28] N. Zaluzec, "ANL TPM/AAEM Collaboratory." [Online]. Available: http://tpm.amc.anl.gov/