A Framework for Building a Scientific Knowledge Grid Applied to Thermochemical Tables

Gregor von Laszewski,[1,*] **Branko Ruscic,**[2] **Kaizar Amin,**[1,3]
Patrick Wagstrom,[1] **Shriram Krishnan,**[4] **and Sandeep Nijsure,**[1,3]
[1]**Mathematics and Computer Science Division and**
[2]**Chemistry Division, Argonne National Laboratory**
[3]**Department of Computer Science, University of North Texas**
[3]**Department of Computer Science, Indiana University**
[*]**corresponding author**
running title: Knowledge Grid

Contents

1 Introduction **1**

2 Thermochemical Tables **2**
2.1 Active Thermochemical Tables 3
2.2 Benefits of the Active Table Approach 4
2.3 Transforming Information and Experience to Knowledge 4

3 Knowledge Grid **5**
3.1 Open Grid Service Architecture 6
3.2 Developing Knowledge Grids 7
3.3 Mobility 7
3.4 Orchestration 9
3.5 Grid Service Flow Language 9

4 Collaborative Framework **11**

5 Ad Hoc Grid Management **12**

6 Grid Interaction Environments **13**

7 Portal to Active Thermochemical Tables **14**

8 Conclusion **14**

A Framework for Building a Scientific Knowledge Grid Applied to Thermochemical Tables

Gregor von Laszewski,[1,*] **Branko Ruscic,**[2] **Kaizar Amin,**[1,3]
Patrick Wagstrom,[1] **Shriram Krishnan,**[4] **and Sandeep Nijsure,**[1,3]

[1]**Mathematics and Computer Science Division and**

[2]**Chemistry Division, Argonne National Laboratory**

[3]**Department of Computer Science, University of North Texas**

[3]**Department of Computer Science, Indiana University**

[*]**corresponding author**

running title: Knowledge Grid

Summary

An important part of enabling the future information infrastructure is to provide convenient services for application users that make such an environment highly functional and easy to use for the nonexpert. In other words, the scientist should be able to focus on the science but should have sophisticated tools at hand that enable new and enhanced modalities in which science can be performed. We describe a general architecture that provides access to scientific applications. We demonstrate how such services can interact with each other and be reused by the scientific application architect to orchestrate Grid services. Our architecture is implemented by reusing concepts, infrastructures, and middleware from Web and Grid services, as well as from the newest Globus Toolkit and the Java CoG Kit.

We apply our general architecture to a specific application: active thermochemical tables. This application demonstrates a number of interesting and future-oriented uses, such as the need for batch processing, interactive and collaborative steering, use of multiple platforms, visualization through large displays, and access via a portal framework. Besides the innovative use of the Grid and Web services, we also provide a novel algorithmic contribution to scientific disciplines that use thermochemical tables. Specifically, we modified the original approach to constructing thermochemical tables to include an iterative process of refinement leading to increased accuracy. We have designed a variety of access environments including a shell, a Gridtop, and a portal for accessing the set of services provided, which include the display of network dependencies between the reactions a chemist may be interested in and interactive querying of associated species data.

Modalities that need to be supported are the protection of intellectual property, the creation of metadata and information shared with and gathered by the community, and the easy access of the service by the application scientists.

1 Introduction

The study of energy changes that accompany chemical reactions and changes in the physical states of matter is referred to as *thermochemistry.* The knowledge of thermochemical stability of substances is central to chemistry and critical in many industries, since chemical reactions are ultimately governed by thermochemistry. Hence, thermochemistry finds applications in other disciplines such as earth science and engineering, helping scientists to better understand processes such as climate and combustion [1, 2] and thus predict and verify them to a high degree of accuracy.

Until now, the thermochemical data necessary for such calculations has been available only in static table form, and the algorithms to derive accurate model descriptions have been too imprecise to deal with the complex chemical reactions encountered in state-of-the-art laboratory experiments or observed in nature. Our goal is to improve this situation by delivering innovative algorithms to the scientists through an advanced collaborative environment.

Novel modalities of deriving new scientific results can be stimulated by enabling a collaborative environment in which scientists can publish and share their results with others, perform sophisticated calculations that are otherwise not feasible, and integrate newly developed algorithms [3].

The Grid [4, 5, 6] can provide the basic middleware infrastructure for bootstrapping this type of sophisticated collaborative environment. The Grid allows scientists to collaborate even though their resources may be controlled by different domains; access to these resources is enabled through the use and creation of "virtual organizations."

In this paper, we describe a general architecture that provides advanced services that can be accessed collaboratively. Their integration as part of a workflow process enables the creation of services that can be easily reused by the community. Scientists can then concentrate on the science, while application developers can focus on the delivery of services that can be assembled as building blocks to create more elaborate services.

Our paper is structured as follows. First, we present the notion of a "knowledge Grid," transforming information into knowledge. Next, we give introduce the problem domain and the terminology used in thermochemistry that is directly related to the work we perform. We analyze a current process to derive thermochemical tables, one of the most elementary building blocks in thermochemistry. Next we provide an improved technique for increasing the accuracy of this process. We introduce a scenario where our algorithm and the repeated use by the community will result in a highly accurate and elaborate thermochemistry table database. We outline our service-oriented architecture and discuss how services such as security, data transfer, registration, and scheduling assist in assembling such a sophisticated collaborative environment. We conclude our paper by summarizing the current state of the project and listing opportunities for further research.

2 Thermochemical Tables

In this section we provide a minimal introduction to basic thermochemistry. Elementary to the discipline of thermochemistry is *enthalpy* (ΔH_f^o), which refers to the value of energy of a system when it is at constant pressure. The enthalpy relationships involved in examining thermochemical equations are easily visualized by means of enthalpy diagrams, such as that shown on the lowerleft-hand corner of Figure 1.

In this diagram the equations are expressed as a graph with horizontal lines representing different values of the enthalpy. Typically the differences between these values are determined experimentally or can be derived by using thermodynamic laws from other enthalpy values. Values obtained from experiments, however, may contain errors (not shown in the diagram). Changes in the enthalpy are visualized by the distance between the lines. Based on the changes performed, different intermediate states (chemical species) may occur during the transition from one to the other final state. Thus, it is natural to visualize the transition

with directed edges between the states. An alternative graph is displayed on the upper right-hand corner that emphasizes the possible states in which a chemical species (i.e., an ensemble of chemically identical molecular entities that can explore the same set of molecular energy levels on the time scale) can occur.

In traditional thermochemical tables the enthalpies of formation (ΔH_f^o) are developed with the help of an elaborate *sequential process*. In each step a new species is added while all ΔH_f^o values determined in previous steps are frozen. The enthalpy of the new species is determined at one temperature T from interconnecting measurements that are limited to those species already defined in previous steps. The temperature-dependent functions, $C_p^o, S^o, (H_T^o - H_o^o), \Delta H_{fT}^o, \Delta G_T^o$, are developed by determining the partition function $Q, lnQ, T\partial(lnQ)/\partial T$, and $T^2\partial^2(lnQ)/\partial T^2$ from the species-specified quantities and the newly selected single temperature enthalpy [7].

The sequential process follows a *standard* order of chemical elements: $O \rightarrow H \rightarrow halogens \rightarrow noblegases \rightarrow chalcogens \rightarrow pnicogens \rightarrow carbonperiod \rightarrow etc.$ For every chemical element introduced, the sequence starts at the standard state for that element, for which $\Delta H_f^o = 0$ by convention. However, enthalpies of formation have complex hidden dependencies. These dependencies are backwards traceable, albeit with considerable manual effort, and in practice are not forwards traceable at all.

This sequential approach of introducing new enthalpies has several disadvantages. In particular, it results in ΔH_f^o and error bars that do not properly reflect the global relationships implied by the species-interconnecting data used. The values and error bars reflect, at best, only local relationships to nearest neighbors. A cumulative error is introduced based on the frozen enthalpies in previous steps. Furthermore, the hidden relationships in conventional tables produce thermodynamic tables that are static in nature. Proper update with new knowledge is nearly impossible because forward relationships are nontransparent; hence, updating one species may improve the quality locally but increase the global inconsistencies.

We present a new approach, based on active tables, that circumvents these disadvantages.

2.1 Active Thermochemical Tables

The active table approach treats the information in the species-interdependent data from the viewpoint of a thermochemical network [7, 8]. Every vertex (node) of this network represents the enthalpy of formation of one species. The species-interdependent data define the topology of the network graph by providing the edges (links) in the network. Competing measurements provide multiple (parallel) links in the network. The relational information define the topology maps onto a set of equations, with the enthalpies of the involved species as unknowns, the stoichiometry of the reactions defining the coefficients, and the measurements and their error bars providing the free elements. Since the number of equations regularly exceeds the number of unknowns, the system is overdetermined. The best solution to the network is obtained in two steps.

The first step statistically analyzes the network with the goal of checking the error bars of individual data items for consistency. All inconsistencies are identified and proposals for their resolutions are generated. The resolutions are generally carried on through incrementing the error bars of the offending data items to their statistically indicated values, and the

analysis is repeated. During subsequent steps the error bars of the offending data items usually oscillate up and down until self-consistency across the whole network is reached. The analysis is carried out in both unweighted and error-weighted space.

The second step occurs once the values and error bars of data items achieve consistency across the whole network. This step consists of finding simultaneously the optimal solution to all nodes by χ^2 minimization (in error-weighted space). The species-specific data is used to prepare the network for analysis by re-expressing all data items as reaction enthalpies at one common temperature. Once the optimal solution to the network has been found, the species-specific data is used to calculate the partition functions and hence develop the temperature dependence of the standard thermochemical functions. We term this approach *active themochemical tables*.

We note that, as opposed to the simultaneous solution described above, in the traditional sequential approach the nodes are solved one at a time in a prescribed sequence. Each of the steps corresponds to selecting a particular path in the network leading from solved nodes to the next node, and ignoring all other possible paths. Both the fact that an overdetermined system of coupled equations is solved for one unknown at the time and that some of the equations are ignored contribute to the lack of global consistency in traditional tables.

2.2 Benefits of the Active Table Approach

Unlike the quantities found in a traditional table, the thermochemical quantities (and their error bars) obtained from the active table properly reflect the globalism of relationships implied by the underlying thermochemical network. All values and error bars are consistent in a global sense. An active table allows rapid update with new measurements (and possibly calculations) by globally propagating the new information through the table. Quality and integrity of the table are protected throughout the updates by error analysis, which runs in both directions: the error bar of the new experiment may shrink error bars in the table; however, the error bars of other experiments in the table might challenge, by means of the statistical analysis discussed above, the error bar assigned to the new experiment. In addition, an active table allows "what-if" scenarios and tests. Such tests provide a critical evaluation of the tested data and its impact on prior thermochemical knowledge or, if the tests correspond to a new experiment, provide feedback on sensitivity of the network to various measurements. The approach also has potential to become an interesting learning and education tool. An active table can provide a ranked list of links that are missing or are weak from a statistical viewpoint; in particular, it can provide pointers to the most useful new experiments or calculations to be conducted.

2.3 Transforming Information and Experience to Knowledge

Thermochemical data is available numerous databases and electronically or published in academic papers. We are faced with some major challanges. First, the update frequency of the thermochemical data is slow and prevents the od hoc integration of new results derived from state-of-the art experiemnts and calculations due to a lengthy publication cycle. Second, the information is presented in a non standard form making it difficult to retrieve it

automaticaly. Third, the number of publications makes it impossible for a single scientist to create such a database. We must be able to selectively include or exclude data to support what-if scenarios, or exclude unwanted data. Hence, we will be able to derive predictors for new, as well as verifiers for recorded reactions.

To enable such new use modalities we need an infrastructure that is conducive to support the scientists' ambitious quest. In order for a researcher to select trusted and wanted contributions the data must be annotated with meta-information. A framework and tools must be avialable in order to assit the organization of the experienceknownledge collectively avaialable within the scientific community. This tools will assit in the transformation from individual experiences into collective knowledge that can be immediately disseminated back into the community. Hence, our environment must fulfill the following requirements:

Collaboration: the environment must support the interaction among scientists in geographically dispersed locations.

Security: the environment must protect from the loss of intellectual property and allows restricted access to the data and compute resources.

Standardization: the environment must enables the scientist to use the tools in a straightforward fashion.

Adaptivity: the environment must be flexible to future changes based on hardware and software.

Dynamicity: the environment must allow the creation of transient services to enable ad hoc collaboration and use of other application services.

Extensibility: the environment must allow the creation of new capabilities and services through the availability of an orchestration framework.

These requirements are shared with many other scientific disciplines, and a large amount of research has been performed in the past few decades to develop frameworks that support such requirements. Our own research efforts in Grids show that the development of advanced services based on our previous research will address many of the requirements by integrating our knowledge deriving algorithms into a service based architecture.

3 Knowledge Grid

The term Grid computing [5] is commonly used to refer to a distributed infrastructure that promotes large-scale resource sharing in a dynamic multi-institutional virtual organization. Just as the electric power grid provides pervasive access to electric power, the computational grid provides ubiquitous access to a large collection to compute related resources and services [4].

Typically such services focus on navigation, storing, and transferring information [9, 10, 11]. A knowledge Grid is the next important step in Grid computing. Like the computational Grid, the knowledge Grid allows the effective management of resources ranging from a personal desktop computer, to resources put together in an ad hoc fashion, to virtual resources shared by a community. Also similar to the computational Grid, the knowledge Grid supports interactions among individual scientists performing research and experiments

on their own, groups, of scientists working together on a problem, and large communities. What distinguishes the knowledge Grid, however, is that it adds to the traditional information services *experience* and *knowledge* services. In Figure 2, we depict the integration of these services, resources, and communities to a knowledge Grid.

An experience service adds additional value to information by integrating the concept of information predictors, a knowledge service is much more complex in nature and uses more sophisticated methods derived from (for example) the artificial intelligence community to build connections between different contextual information and experience services to derive knowledge. In Figure 2 we depict the basic interplay between our different concepts that provides us with a roadmap toward the development of a knowledge Grid. We can bootstrap our environment from an information service-based to a knowledge service-based Grid.

3.1 Open Grid Service Architecture

Our framework for knowledge Grids will is based on standards under development by the Grid computing community to provide a solid foundation for our current and future work.

The Grid architecture identifies and solves a wide range of problems that arise as a result of large-scale resource sharing among separately administered institutions. Within the Internet and Grid communities we observe most recently the trend to base the next generation of architectures on the Web services model.

A Web service is a platform-indepedent software component that is described using a service description language [12] and provides functionalities to publish its interfaces to a service registry. Published services can be discovered by querying the registry, and invoked using appropriate binding protocols [12]. This discovery and binding process allows clients to interact with services that are not available locally but are accessible through a distributed networked environment such as the Grid envisions. Therefore, a service oriented architecture lends itself to enabling interoperability across heterogeneous platforms and exposes all or part of its applications on different machines, platforms, and domains.

The Grid community is enhancing the Web service model to address issues such as transient services, service lifetime management, and event notification [13], while also focussing on problems and solutions address important issues such as security, resource management, and resource discovery [14, 15].

Web services address the discovery and invocation of persistent services, whereas Grid technologies need to tackle issues related to creation, discovery, and destruction of transient service instances. Hence, the Open Grid Services Architecture (OGSA) [16] was developed to provide an extensible framework of services that combines the realms of Web services and Grid technologies. OGSA introduces the notion of Grid service: a possibly transient Web service that provides a set of well defined interfaces and that follows some standard conventions [16]. In summary, OGSA provides a framework that supports

instance creation through Grid service factories that can request new reliable transient service instances. OGSA provides a mechanism to individually identify these instances by associating state information with each service instance.

lifetime management to build transient service instances that can negotiate with the service Factory at the time of their creation duration and lifetime behavior. It may support

soft-state lifetime management, thereby avoiding the need for explicit destruction of service instances.

registration and discover to allow the registration and discovery of transient service instances. OGSA provides an efficient query mechanism for the client using a search operation for Grid service against the meta-data maintained by that registry to describe the services.

notification to establish a publish/subscribe style of notification mechanism between service providers and service requesters, thereby supporting asynchronous peer-peer style of communication.

security to handle the required authentication and authorization policies defined by different application and domain administrators.

3.2 Developing Knowledge Grids

OGSA services provide a foundation for sharing resources and services across virtual organizations. The addition of knowledge services to information and knowledge services provides another essential capability for effective Grid computing. Nevertheless, we need a number of other critical capabilities for our knowledge Grid:

- Orchestration - to express interactions between services thereby enabling *dynamic composition* of new and more capable services from existing ones.
- Mobility - to deal with the disruptive,sporadic, and ad hoc nature of the Grid while addressing service *mobility, autonomy, and self-healing* with quality assertions expressed through policies

We believe that all three concepts orchestration, mobility, and knowledge must be addressed to build the next generation Grid middleware reusable as part of sophisticated knowledge-driven applications. This integrated architecture is depicted in Figure 3.

In the next two sections we elaborate on the ideas of mobility and orchestration.

3.3 Mobility

As part of our framework we propose advanced Grid services that deal with failures and recovery. Such services are essential to enhance the current generation of Grid services that can respond to the disruptive behavior common in Grids. These, augmented services will contribute to the reliability of the Grid. While augmenting such services with quality and movability assertions we will be able to build a subset of our services to be migratable and autonomous Grid services that act in behalf of a user or a virtual organization. Based on our experience in developing hosting environments for mobile agents, we identified the following set of basic services that are instrumental in the creation of such a dynamic environment:

- An autonomous application factories, which creates jobs that may be executed on remote resources and is also allowed to be moved to a different location without losing endpoints to the tasks started within the factory. An application factory can be augmented with a set of engines that increase the functionality. Such engines include logging, constraint and policy, event, and reliable messaging.

- An autonomous logging engine, which is a basis for almost every service to allow logging of events, application check-pointing, and the creation of fault tolerant services. A mobile application factory may instantiate a logging service. The logging data may move appropriately based on the autonomous behavior of the initiating service.
- An autonomous constraint and policy engine which may be associated with an execution broker and is used to verify constraints within the current execution environment. This includes authentication and authorization augmentations.
- An autonomous event engine, which is needed for performing event notifications based on the state of a variable, datum, or service.
- An autonomous messaging service, which communicates between services in a reliable fashion.

The services that we list here are based on elementary Grid services such as those provided by the Open Grid Service Architecture, but enhance the existing services with a methodology of enabling mobile agents through autonomous behaviors. Thus, each service can be augmented with semantically defined annotations that determine the behavior in the context of Grids. Several of these services need to engage in contract and use policy verification before migration is enabled to a different hosting environment within the Grid. To this end, we will use and enhance the underlying Grid security mechanisms to enable the possibility of engaging in contract negotiations before and during runtime execution of the application. Furthermore, we will define a set of use policies and contracts that can deal with the dynamic nature of the Grid. Based on these services and services that we define in other work (third-party file transfer, general information service, job execution service, job monitoring service, and resource management service), we develop more advanced services that hide much of the underlying functionality that is nonessential for the middleware developer.

To enable such sophisticated services, we need to develop a number of elementary services. This allows us to provide a flexible design while gradually integrating new services into the framework (see Figure 4).

Grid Broker Service to deal with large numbers of calculations that are involved with future large-scale reactions and their real-time requirement for allowing interactive use [17, 18].

Grid Workflow Service to enable the interplay of Grid services through workflow descriptions [19, 20].

Grid Execution Factories to enable the execution of programs in a Grid while instantiating them in a hosting environment and making their results accessible to other services (the Globus Toolkit [21] provides such services in Java and C) [19].

Grid Monitoring Service to monitor the state of the hosting environment so that feedback to Grid services is provided, enabling the environment to react to state changes [22].

Grid Migration Service to be able to migrate services and jobs executed with a Grid Execution Factory Service to a location that is better suited, based on performance and quality of service descriptions and policies [23].

Grid Logging Service to log and checkpoint services in order to enable migration and fault-tolerant behavior.

Grid Self-Healing Service to determine how and when it is necessary to change the dynamically instantiated Grid workflow applications (including preventive measurements such as service replication, service migration, service checkpointing, and service monitoring).

Collaborative Steering Service to collaboratively create data, thoughts, and ideas that will lead to new scientific findings [24].

3.4 Orchestration

We need to build an orchestration framework to unleash the power of the Grid services model. Orchestration includes two concepts: specification of dependencies between services and the specification of semantic dependencies that determine that interactions between the services.

Both concepts have received considerable interest recently in industry and academia.

We have contributed to this research field with our development of (1) a Grid Services Flow language [25], which led to the development of the first XML based language that is able to express elementary orchestration between Grid services based on OGSA alpha and (2) a basic workflow management infrastructure as part of the Java CoG Kit [26, 19].

Efforts from industry include the Web Services Flow Language [27, 28] and XLANG [29], which have been most recently combined into a new technology called Business Process Execution Language for Web Services (BPEL4WS) [30]; the definition of DAML-S [31]; Web service Orchestration Interface (WSCI) [32]; and the Business Process Modeling Language (BPML) [33] a metalanguage for modeling business processes.

Many Grid-related workflow systems have been developed, including Webflow [34], DAGMan [35], Unicore [**?**] , and XCAT [36]. Webflow is one of the earliest successful systems providing workflow management for Grid like systems. DAGMan is a meta-scheduler for Condor [37] that manages dependencies between jobs. Some of the tools mentioned above use direct acyclic graphs as the underlaying workflow data structure. However, the concept of using a directed acyclic graph to represent a set of programs where the inputs, outputs, and the execution are interdependent cannot be applied to describe the dependencies between the Web services. Hence, the XCAT Application Factories address workflow issues for Grid-based components within the Common Component Architecture (CCA) [38] framework. XCAT allows components to be connected to each other dynamically, making it possible to build applications in ways not possible with the standard Web services model.

3.5 Grid Service Flow Language

Since the current standards do not address the features implicitly included in OGSA, a framework is needed that allows for orchestration of such services with regards to instance creation, lifetime management, registration and discovery, notification, and security, while at the same time fulfilling a number of high level requirements that allow us to formulate, deploy, instantiate, and automate workflow orchestration.

It will be essential to address these issues within a Web service orchestration language.To create such a mechanism, we must not only describe the order in which these services and

their methods execute but also present a way in which such an agglomerate can export itself as a service. In this context, we define the term "orchestration" as a set of rules that define the interactions between a set of services in order to be composed into a meta-service. The set of rules we must deal with include contracts, exception handling, process control, asynchronicity, and lifetime management. Scalability, mobility, and autonomic behavior are additional concepts to be addressed while implementing our workflow engine in the concept of a knowledge Grids.

Just as Web services aim to do, the Grid workflow specification should allow specific activities implemented by individual services to be exported as activities of the workflow. It should also allow the exported activities to trigger a chain of other activities. Current technologies such as WSFL address these issues effectively.

Hence, we incorporated features presented by WSFL into the Grid Services Flow Language we designed. The specification is rich enough to describe the workflow such that the WSDL for the workflow entity (henceforth referred to as a *workflow coordinator*) can be auto-generated from the specification. The workflow coordinator must be able to handle the methods that have been dynamically exported as a composition of the various activities of the workflow, and handle them in such a way that clients can access them using the same standard tools that they use to deal with the individual services. This is an important requirement for recursive composition of services.

Despite the fact that it is theoretically possible to define peer-to-peer interactions between Web services that are part of the workflow in languages such as WSFL (via *plugLinks*), it is not feasible because solicit-response and notification operations are not fully defined in WSDL 1.1. There are multiple interpretations of these operations in the Web service community, and there is considerable debate about their removal in the forthcoming WSDL 1.2 [39] specification. As a result, and as has been pointed out by [36], existing Web services define their workflow in such a way that the workflow engine has to intermediate at each step of the application sequence, as shown in Figure 5. In business-to-business communication the workflow engine does not end up being a bottleneck because there may only be a moderate level of data transmission between the services. For Grid-based services, however, exchanging large amounts of data is the norm. Having a central workflow engine relay the data between the services would be a bad idea in this case. The workflow specification needs to be able to allow communication between the services, as depicted in Figure 6.

As we mentioned, OGSA adds extensions to WSDL in order to address Grid-specific needs. It addresses communication between Grid services by using notification sources and sinks, which allow services to carry out asynchronous delivery of messages between each other. GSFL must provide a mechanism to connect notification sources and notification sinks, thus obviating the need for the workflow engine to mediate at every step. Additionally, OGSA uses *registries* and *factories* for locating and creating Grid services, respectively. These must be appropriately handled by GSFL.

It is conceivable that certain Grid services in the workflow will not be executing while others are. One reason may be the fact that the services that need to execute earlier run for weeks; another reason may be that the service that executes later needs data from the former to bootstrap itself. The Grid workflow specification should be able to handle these situations. Additionally, it should also be able to handle instantiation of the individual

Grid services on a per method or a per workflow instance basis. If the Grid services are instantiated on a per workflow instance basis, certain activities exported by the workflow may not function because a certain Grid service may have run to completion or may not have been instantiated yet. In such a case, certain ordering has to be imposed on the exported activities, such as the one proposed by WSCL [40].

We have designed the Grid Services Flow Language based on the XML-schema based language allowing the specification of orchestration and workflow descriptions for Grid services in the OGSA framework. It has the following important features, which are explained in more detail in [25].

- *Service Providers*, which are the list of services taking part in the workflow;
- *Activity Model*, which describes the list of important activities in the workflow;
- *Composition Model*, which describes the interactions between the individual services; and
- *Lifecycle Model*, which describes the lifecycle for the various activities and the services that are part of the workflow.

Together these features are sufficient to support our requirements to express workflow and orchestration within our knowledge Grid.

4 Collaborative Framework

To support our need for collaboration, we have defined a set of collaborative services that we term *Open Collaborative Grid Services Architecture* (OCGSA), which contains a set of common components that can be easily customized for individual applications. The OCGSA services enable users to form ad hoc collaborative groups by interacting over a set of predefined notification topics. It provides appropriate lifetime management for individual groups, offers an advanced discovery mechanism for service instances, and establishes sophisticated security mechanisms at different levels of the application [41, 24].

Rather than redefining metadata information for every collaborative Grid service, we introduce the notion of a *collaborative Grid service*: a Grid service with a set of predefined metadata elements and behaviors. In analogy with the Grid Service port type [42], we define a *collaborative Grid service port type* that provides base interfaces required for a collaborative Grid service instance. For example, it defines metadata with regard to the creator, name, description, and current members of the group, as well as the level of event archiving desired. Additional interfaces are designed to handle security policies at different levels of the application (application level and group level), basic community chat facilities, an adequate presence management for users, and capabilities for interactively controlling and steering the application. We are formalizing these concepts in a *Collaborative Grid Service Descriptive Language* (CGSDL) by which a developer is able to predefine a set of notification topics that will be supported by collaborative Grid services. The collaborative Grid description language is an extension to the *Grid Service Description Language* (GSDL)

defined by OGSA and is used to describe details of a collaborative Grid service. A code generator will generate necessary stubs for this definition, while allowing application-specific details for each notification topic be implemented by the developer.

To archive the data for later queries and reuse, we have developed a sophisticated archiving service that functions more than a simplistic event database. The event archiving service in the OCGSA framework is designed as a Grid service, thereby efficiently handling the lifetime management policies of individual archives that can be set and manipulated by creators of corresponding group instances. Further, the event archiving service will implement the same security policies as implemented by the collaborative Grid services, hence forcing appropriate authorization policies at the application level and individual group level. This strategy ensures that a malicious user lacking the proper credentials, who was denied access to an active group earlier cannot have access to the communication archives when the group is off-line.

Similar to discovering a collaborative Grid service, it is extremely important to efficiently and conveniently discover event archives. On fundamental level, the archiving system must support a *query-by-name* style of information retrieval indicating whether an archive for a particular group exists. Support must also be built to enable advanced queries of meta data values of individual group archives similar to that of the OCGSA registration component. Hence, end-users will be able to discover archives hosted by a set of archiving services based on complex queries regarding group names, lifetime validity, security authorization, and similar criteria.

To implement the functionality mentioned above, OCGSA provides an Event Archiving Grid service that provides an interface to XML:DB. Hence, the Grid service can take care of the security and lifetime management issues, and the XML:DB (NXD, native XML database) provides the functionality for the data storage and retrieval using XPath queries. The XML:DB can be embedded within the Grid service or can be located on an individual machine. The event archiving service can be considered as a specialization of a generic data storage Grid service that provides a Grid service interface to a native XML database (NXD) implementing all the generic operations required to manipulate the XML database that can store any XML document including event logs.

5 Ad Hoc Grid Management

An important success of Grids will be based on their ease of use and intractability in the daily activities of scientists. Thus it is important that software and methodologies are developed that allow scientists to use the Grid in an easy and seamless fashion. The Java CoG Kit provides the ability to develop easy to use and deploy components.Such a strategy, although costly in the development has a great payoff due to increased usability. Thus we have decided to follow the strategy demonstrated by the Java CoG Kit and investigate in the development of a number of services that allow the deployment and use of the collaborative framework by the non expert. To these services not only belong our execution factory, but also services that directly relate to the instantiation of a virtual organization on behalf of a user, group, or community. Additionally, we prototyped a tool that can wrap any command line oriented

program as Web service (Figure 7). Once the OGSA release is finalized, we will be able to deliver such a command line to Web services converter also for OGSA.

This will bring us one step closer in building, and managing communities and resources in an organized but also ad hoc fashion.

6 Grid Interaction Environments

To interact with a knowledge Grid, we need to developed a variety of components with different user communities in mind. As the current generation of Web browsers is rather limited in functionality, we will be able to provide a simple portal infrastructure for elementary interactive uses based on simple query mechanisms and page reload. We have developed such an interface for our application domain based on the Jetspeed framework, which allows users to can interact with the application Web services via portlets. Portlets are applications running inside a Web-based portal framework. It provides a mechanism for individual look and feel as well as the choice to integrate new components in one's personalized window to the knowledge Grid. Choice of a portal framework offers many advantages. A variety of scientific applications can be accessed from a single portal. Hence, it provides a uniform visual interface and, in most cases, a single sign-on. Each user can easily customize a personally owned portal page to include only the commonly used applications. The portal can be accessed with only a Web browser, thus relieving the scientist of any complicated software installation procedures. It can even be used from places such as libraries and airports, where a computer connected to the Internet may be available. The portal we have built for the CMCS project allows scientists to access the query service and reaction graph service as described earlier. In addition, the portlet features an easy-to-use JavaScript menu that lets the user change Active Chemical Tables application parameters and send commands to the application. Our design include s WebDAV server so results can be stored with versioning information. A typical user session looks like following:

1. The user logs into the portal.
2. The user chooses the ATcT portlet from the list of available portlets and adds it to her portal page. This needs to be done only on the first visit to the portal. The portal remembers the users choice.
3. The user communicates through the portlet to query thermochemical data for a chemical species, for example, carbon. This causes the portlet to communicate with the query Web service, sending it the species name. The Web service passes it on to the AT application, retrieves the result from the application, and sends it to the portlet.
4. Thermochemical data may be unavailable or limited, because of several reasons. The species may not be in ATcT database. Enthalpy data may need to be provided by the user. A reaction network may need to be constructed.
5. The user gathers the necessary input data on her WebDAV workspace and then uses the portlet to retrieve this data and send it to the a query service, which passes it on to the AT application.
6. The user instructs the ATcT application to solve the reaction network by choosing the "solve" command from the portlet menu.

7. A new query is submitted to retrieve a complete result.
8. The result is stored into a personal maintained WebDAV workspace, which can be integrated with other Web Dav services..

Nevertheless, we observe often that this interface is not sufficient and a tighter integration with the users desktop is desired. We term such a Grid-based thick client that may use the newest Web and Grid protocols to communicate with the backend services a GridTop. As part of such GridTops we have developed command shells and visual components that provide sophisticated user interfaces on the client side, but can be deployed and upgraded easily through Web technologies. Additionally, we have also developed a portal that is based on jetspeed technology. The user interacts through the portal (or other interface) with the sophisticated environment, so that scientists may concentrate on the science and not the environment [4, 20, 17].

7 Portal to Active Thermochemical Tables

As part of the SciDAC CMCS project, we have implemented the concepts and components described in the proceeding section in a prototype knowledge Grid focusing on the active thermochemical tables. We also enhanced the original concept to enable steering and activating activate components in collaborative fashion while using access control through collaborative groups. We use the Grid security infrastructure it is possible to provide secure application sharing and control.

In Figure 8 we outline several technical components that are described in more detail in [3] to illustrate how these services interact with each other. They are accessed through a portal and use a shared data storage that can be controlled by individuals, groups, and communities. We see that knowledge is created with the help of sophisticated applications, and supported through a sophisticated Grid infrastructure supporting the vision.

8 Conclusion

Scientific applications are complex and produce an ever increasing amount of information. To deal with this information, we need a framework that can be expanded and deal with future demands. The Grid services framework builds a foundation for such an infrastructure. Nevertheless, we need to provide advanced services that transform the large amount of information and experience available in domain-specific and interdisciplinary scientific into knowledge. To this end, we have proposed a knowledge Grid that requires sophisticated domain-specific algorithms, collaborative, and mobile Grid services, as well as Grid service flow language that takes into account specific requirements of the newest standards defining Grid services. We have developed a prototype environment that uses a novel algorithm such as active thermochemical tables. We demonstrated this environment at SC'2002.

Acknowledgments

This work was supported by the Division of Chemical Sciences, Geosciences and Biosciences, Office of Basic Energy Sciences, and by the Mathematical, Information, and Computational Science Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38. Funding for the Globus Project research and development is provided by DOE, DARPA, NASA, and NSF. The work is performed with help of the SciDAC Collaboratory for Multi-scale Chemical Sciences (CMCS) and the SciDAC Java Commodity Grid project sponsored by the Department of Energy. We thank Jim Myers, Larry Rahn, David Leahy, David Montoya, Karen Schuchardt, Carmen Pancerella, Christine Yang, Ian Foster, the other members of the SciDAC CoG Kit project, the SciDAC CMCS project, and the Globus Project. The Globus Toolkit and Globus Project are trademarks held by the University of Chicago.

Bibliographies

Gregor von Laszewski is a scientist at Argonne National Laboratory and a fellow at the University of Chicago Computation Institute. He specializes in Grid computing. He is the principal investigator of the Java CoG Kit, which provided the first defacto standard for accessing Globus through Java. He obtained his B.S. and his M.S in computer science at University of Bonn, Germany. In 1996, he received a Ph.D. form Syracuse University, where he developed a loosely coupled metacomputing environment allowing seamless access to several supercomputing centers through workflows. He has published more than forty papers and technical reports in the area of distributed and Grid computing.

Branco Ruscic is a scientist in the Chemistry Division of Argonne National Laboratory. His current interests are in the spectroscopic and thermochemical properties of free radicals and other transient species relevant in combustion and atmospheric chemistry, using photoionization mass spectrometry and photoelectron spectroscopy with conventional, laser and synchrotron light sources, as well as very highlevel electronic structure calculations resulting in accurate thermochemical properties. He has received a Ph.D. in physical chemistry from the University of Zagreb, Croatia, in 1979. He has had various appointments with the Rugjer Boskovic Institute in Zagreb and the Physics and Chemistry Divisions at Argonne National Laboratory. He has been permanently at Argonne National Laboratory since 1988. He has authored or coauthored approximately one hundred scientific and technical publications and holds one patent.

Kaizar Amin Kaizar Amin is a Ph.D. student in the computer science department at the University of North Texas. He received his B.E in computer engineering from Mumbai University (India) and M.S in computer science from the University of North Texas. His research interest includes adaptive middleware for Grid computing, peer-to-peer Grids, Grid workflows, and Grid mobility using mobile agents.

Sandeep G. Nijsure received his B.E in Computer Science from University of Bombay (India) and M.S in Computer Science from the University of North Texas. During this project he worked as research assistant at Argonne National Laboratory. After graduation he will be working with Microsoft Corporation.

Patrick Wagstrom

Shriram Krishnan [43]

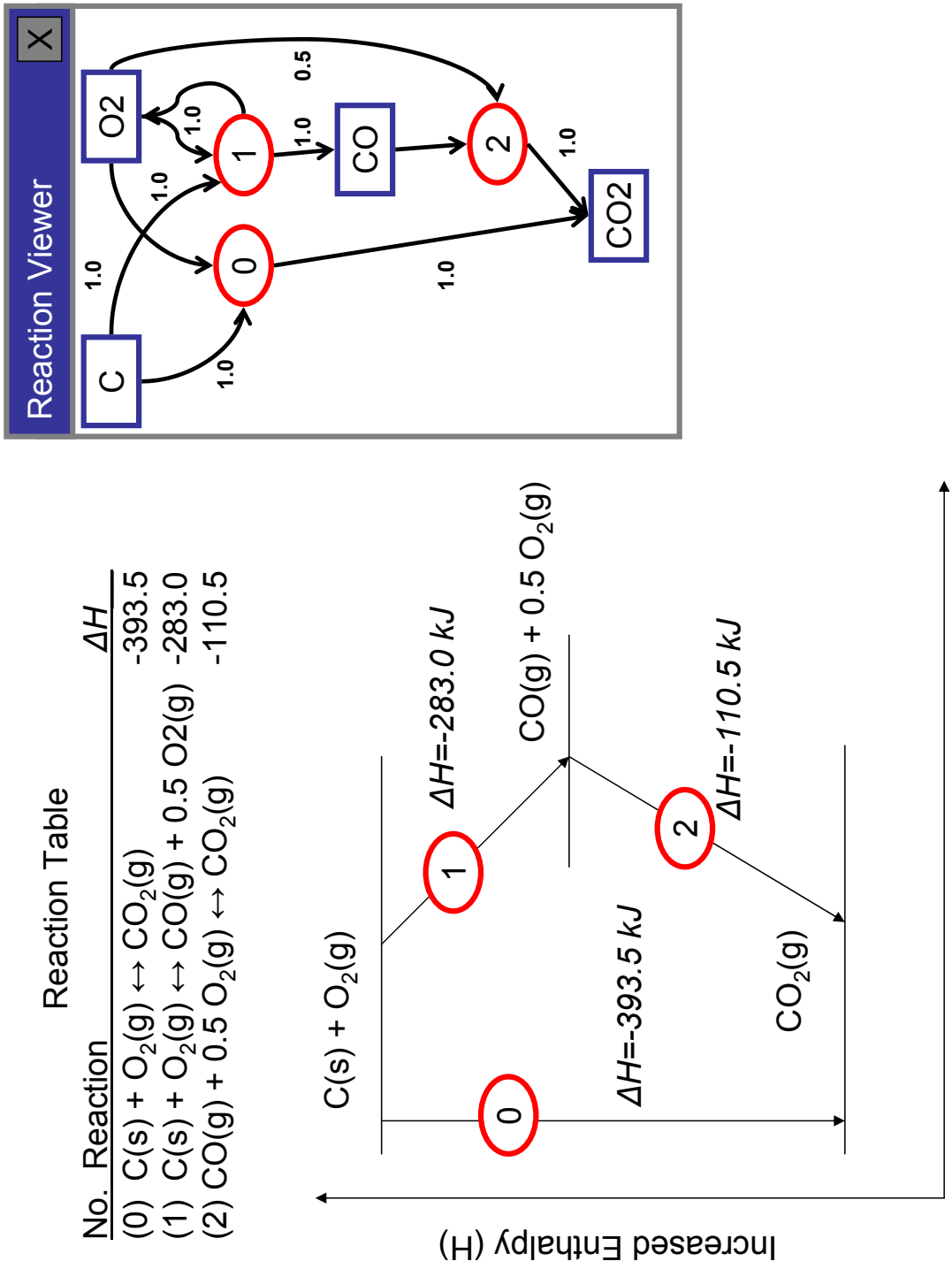

Figure 1: Enthalpy diagrams and thermochemical reaction tables.

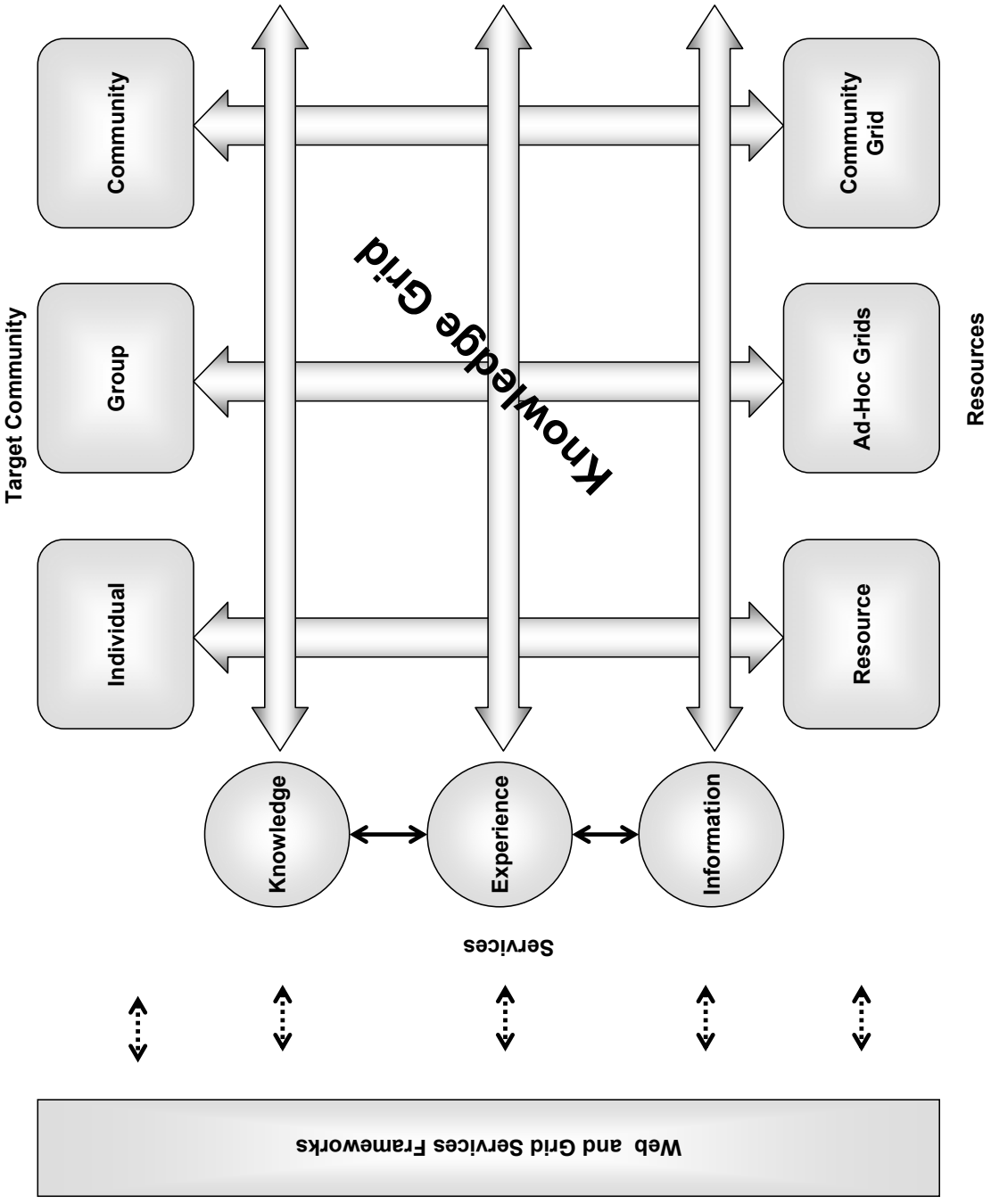

Figure 2: A knowledge Grid is necessary to deal with the large amount of information that is created by interdisciplinary scientific applications, managing resources, and building communities.

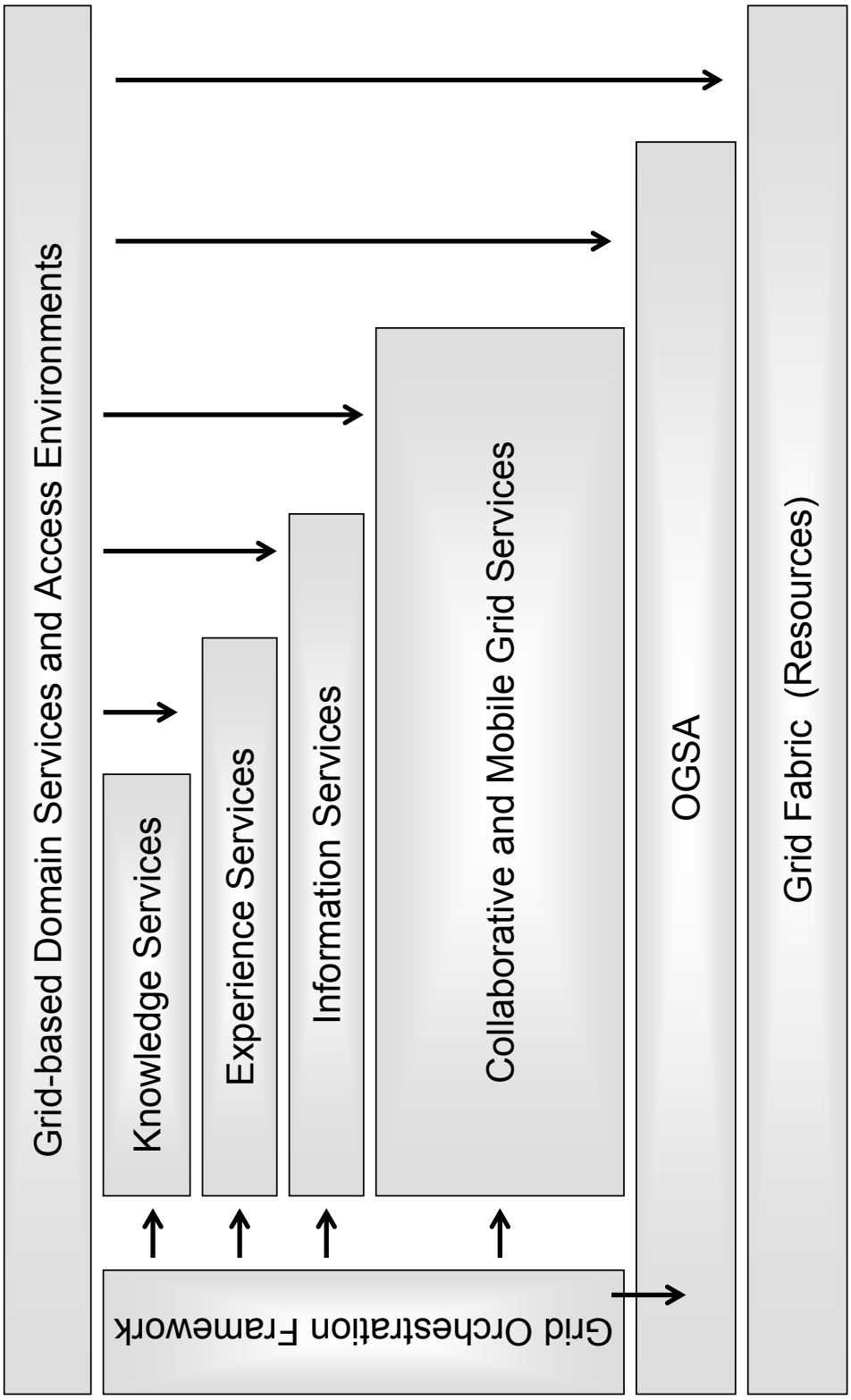

Figure 3: Our general architecture is build on several layers that make it possible to develop advanced services that are mobile and can be used to build ad hoc Grids to support group and community collaborations.

Figure 4: Orchestration is an important part of a flexible Grid middleware infrastructure

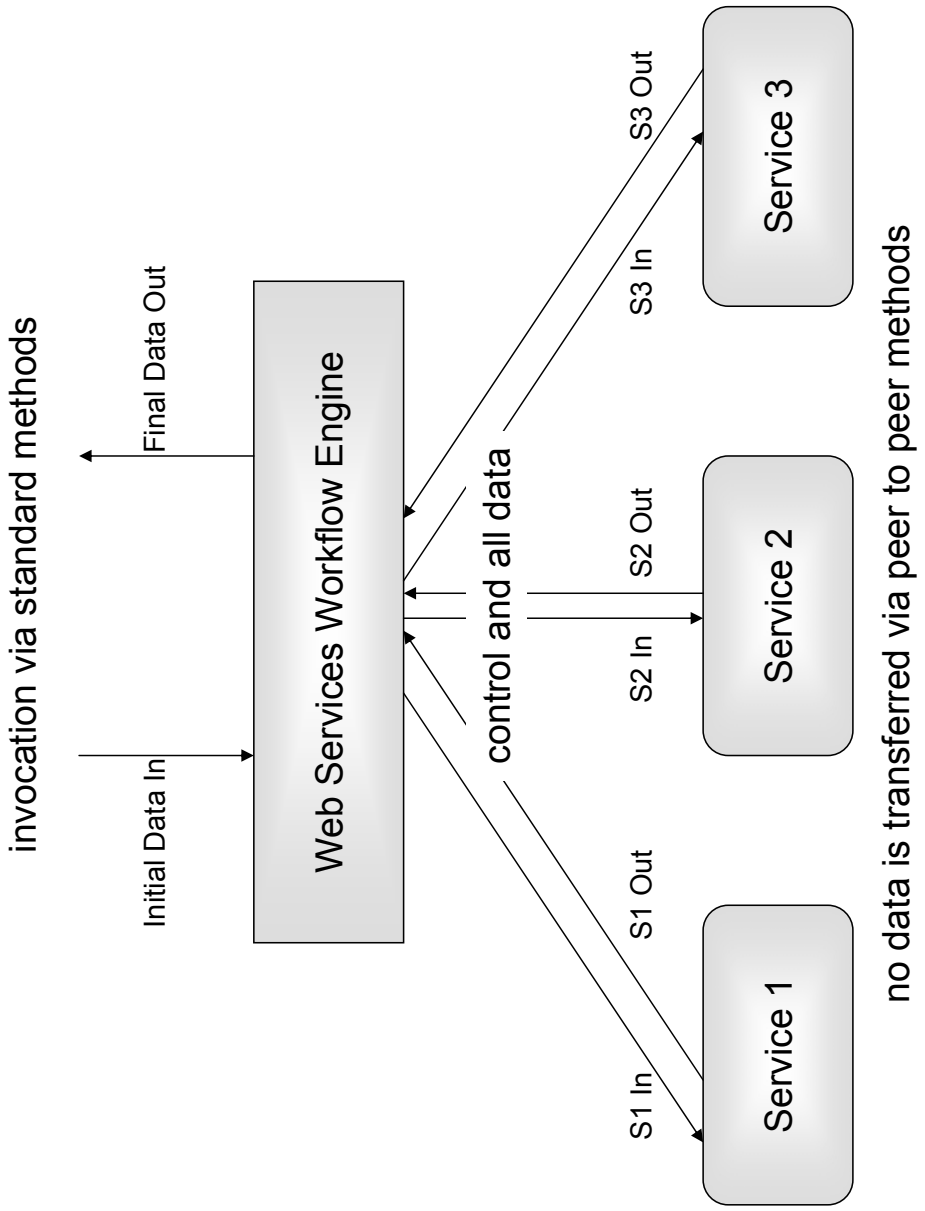

Figure 5: Web services workflow model

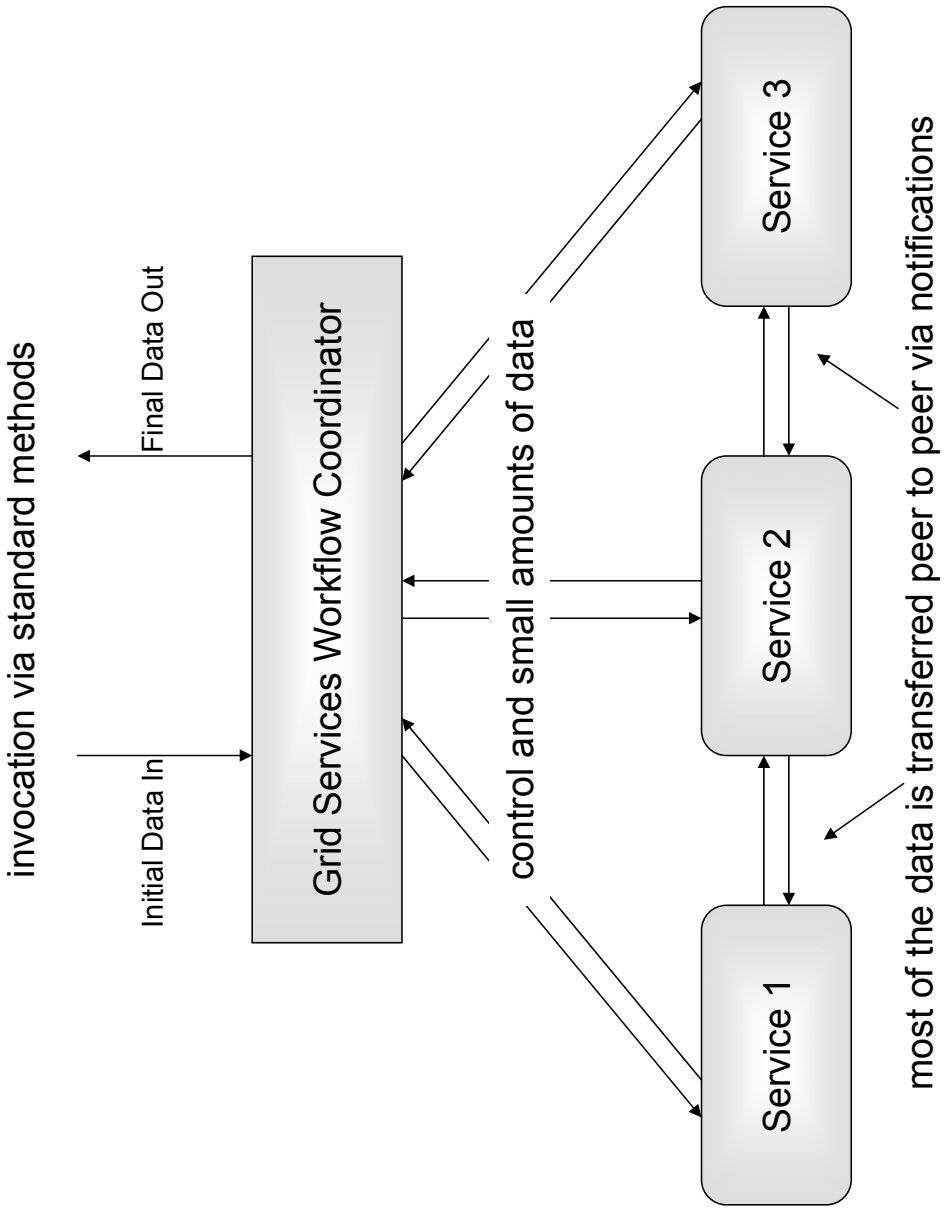

Figure 6: Grid services workflow model

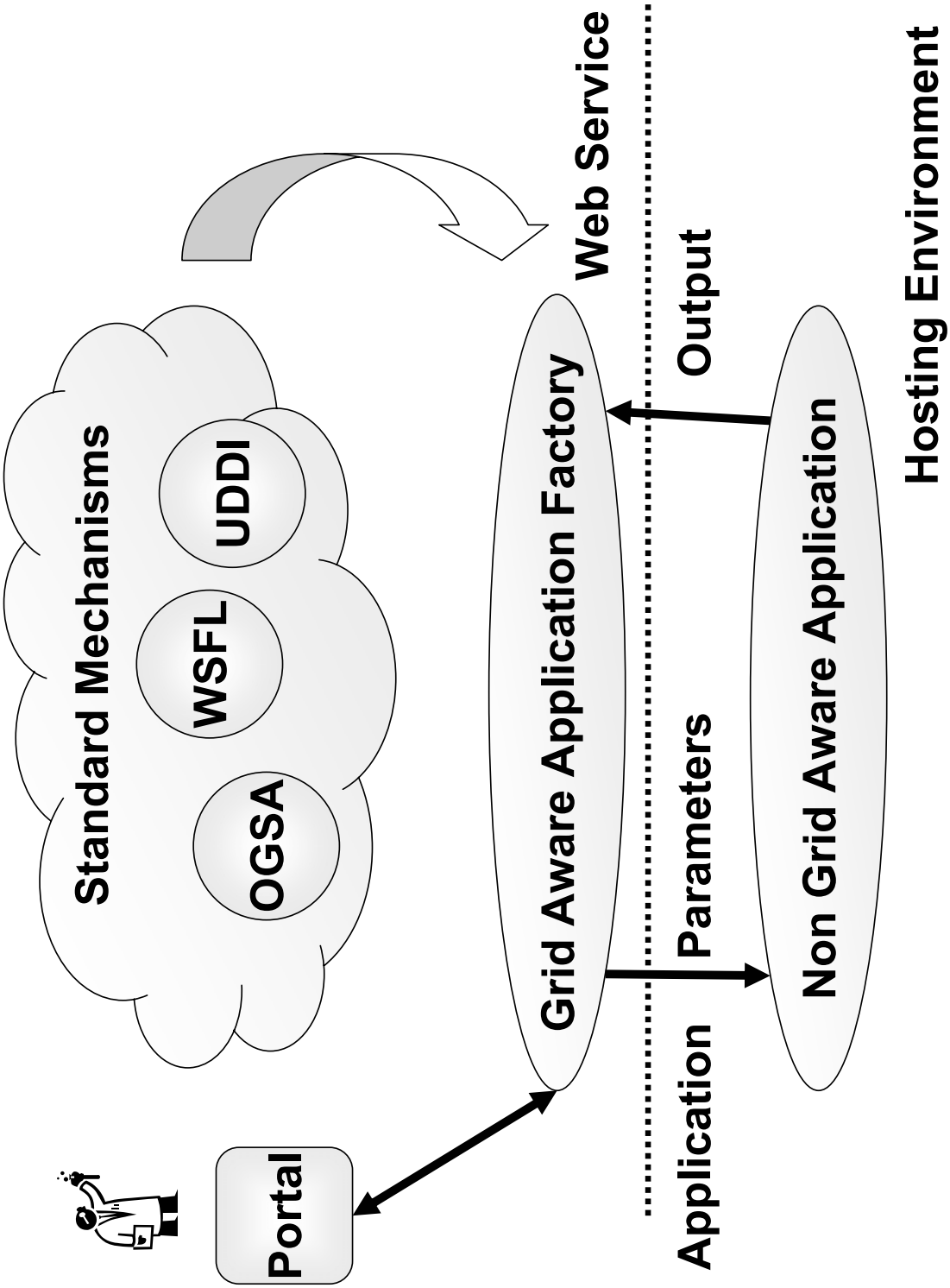

Figure 7: A tool to convert existing command line- and library- based applications is needed to quickly integrate new services into the future service-oriented Grid infrastructure.

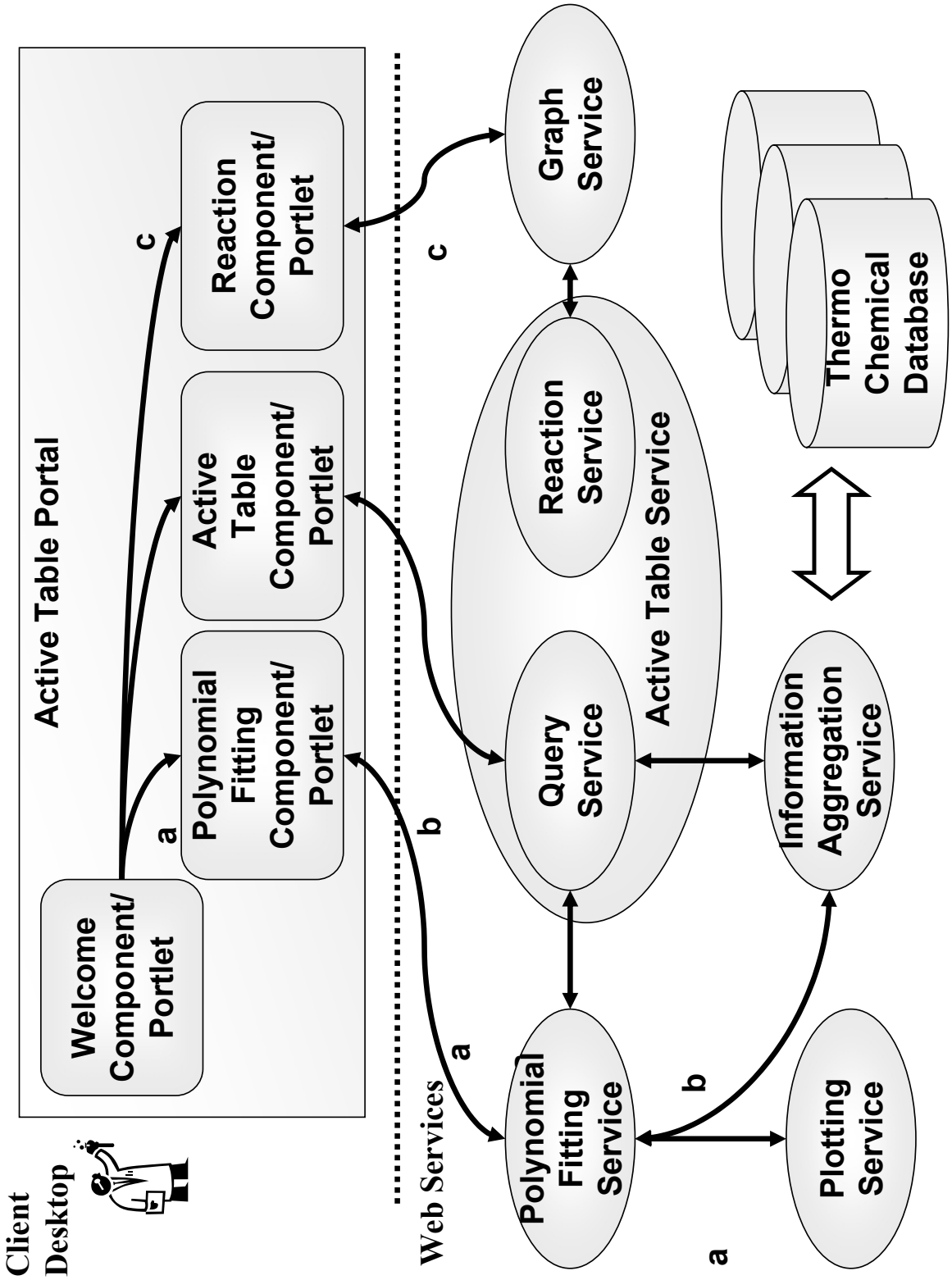

Figure 8: Components of the knowledge Grid for Active Themochemical Tables

References

[1] C. M. Pancerella, L. Rahn, and C. Yang, "The Diesel Combustion Collaboratory: Combustion Researchers Collaborating over the Internet," in *Proceedings of SC99*, Portland, OR, November 13-19 1999, http://www.supercomp.org/sc99/.

[2] M. Frenklach, *Combustion Chemistry.* Springer-Verlag, 1984, ch. 7 Modeling, pp. 423–453.

[3] G. von Laszewski, B. Ruscic, P. Wagstrom, S. Krishnan, K. Amin, S. Nijsure, R. Pinzon, M. L. Morton, S. Bittner, M. Minkoff, A. Wagner, and J. C. Hewson, "A Grid Service Based Active Thermochemical Table Framework," in *Third International Workshop on Grid Computing*, ser. Lecture Notes in Computer Science. Baltimore, MD: Springer, 18 Nov. 2002, pp. 25–38. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/bib/papers/vonLaszewski--cmcs.pdf

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

[5] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications*, vol. 15, no. 3, 2001. [Online]. Available: http://www.globus.org/research/papers/anatomy.pdf

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

[7] B. Ruscic, J. V. Michael, P. C. Redfern, L. A. Curtiss, and K. Raghavachari, "Simultaneous Adjustment of Experimentally Based Enthalpies of Formation of CF3X, X = nil, H, Cl, Br, I, CF3, CN, and a Probe of G3 Theory," *J. Phys. Chem. A.*, vol. 102, pp. 10 889–10 899, 1998.

[8] B. Ruscic, M. Litorja, and R. L. Asher, "Ionization Energy of Methylene Revisited: Improved Values for the Enthalpy of Formation of CH2 and the Bond Energy of CH3 via Simultaneous Solution of the Local Thermochemical Network," *J. Phys. Chem. A.*, vol. 103, pp. 8625–8633, 1999.

[9] G. von Laszewski, M.-H. Su, J. A. Insley, I. Foster, J. Bresnahan, C. Kesselman, M. Thiebaux, M. L. Rivers, S. Wang, B. Tieman, and I. McNulty, "Real-Time Analysis, Visualization, and Steering of Microtomography Experiments at Photon Sources," in *Ninth SIAM Conference on Parallel Processing for Scientific Computing*, San Antonio, TX, 22-24 Mar. 1999. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/papers/vonLaszewski--siamCmt99.pdf

[10] "Griphyn - grid physics network," Web page. [Online]. Available: http://www.griphyn.org/index.php

[11] "http://eu-datagrid.web.cern.ch/eu-datagrid/." [Online]. Available: http://eu-datagrid.web.cern.ch/eu-datagrid/

[12] M. G. Christensen E., Curbera F. and W. S., "Web services description language," 2000. [Online]. Available: http://msdn.microsoft.com/xml/general/wsdl.asp

[13] 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.

[14] S. Graham, S. Simeonov, T. Boubez, D. Davis, G. Daniels, Y. Nakamura, and R. Neyama, *Building Web Services With Java*. SAMS, 2002.

[15] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Open Grid Service Infrastructure WG, Global Grid Forum, 22 June 2002. [Online]. Available: http://www.globus.org/research/papers/ogsa.pdf

[16] "Open grid services architecture homepage," http://www.globus.org/ogsa, April 2002. [Online]. Available: http://www.globus.org/ogsa

[17] G. von Laszewski and I. Foster, "Grid Infrastructure to Support Science Portals for Large Scale Instruments," in *Proceedings of the Workshop Distributed Computing on the Web (DCW)*. University of Rostock, Germany, 21-23 June 1999, pp. 1–16, *(Invited Talk)*.

[18] "Condor home page," http://www.cs.wisc.edu/condor/. [Online]. Available: http://www.cs.wisc.edu/condor/

[19] 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.globus.org/cog/documentation/papers/cog-cpe-final.pdf

[20] G. von Laszewski, I. Foster, J. Gawor, P. Lane, N. Rehn, and M. Russell, "Designing Grid-based Problem Solving Environments and Portals," in *34th Hawaiian International Conference on System Science*, Maui, Hawaii, 3-6 Jan. 2001, http://www.mcs.anl.gov/~laszewsk/papers/cog-pse-final.pdf, http://computer.org/Proceedings/hicss/0981/volume%209/0981toc.htm. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/bib/papers/vonLaszewski--cog-pse-final.pdf

[21] "The Globus Project," June 2002. [Online]. Available: http://www.globus.org/

[22] 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/~laszewsk/papers/infogram.ps

[23] G. von Laszewski, K. Shudo, and Y. Muraoka, "Grid-based Asynchronous Migration of Execution Context in Java Virtual Machines," in *Proceedings of EuroPar 2000*, ser. Lecture Notes in Computer Science, A. Bode, T. Ludwig, W. Karl, and R. Wismüller, Eds., vol. 1900. Munich, Germany: Springer, 29 Aug. - 1 Sept. 2000, pp. 22–34, *(Invited Talk)*.

[24] K. Amin, S. Nijsure, and G. von Laszewski, "Open Collaborative Grid Services Architecture (OCGSA)," in *Euroweb 2002 Conference, The Web and the GRID: from e-Science to e-Business*. St Anne's College Oxford, UK: The British Computer Society, 17-18 Dec. 2002, pp. 101–107.

[25] S. Krishnan, P. Wagstrom, and G. von Laszewski, "GSFL: A Workflow Framework for Grid Services," in *Preprint ANL/MCS-P980-0802*, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, U.S.A., 2002. [Online]. Available: http://www-unix.globus.org/cog/papers/gsfl-paper.pdf

[26] G. von Laszewski, "A Loosely Coupled Metacomputer: Cooperating Job Submissions Across Multiple Supercomputing Sites," *Concurrency, Experience, and Practice*, vol. 11, no. 5, pp. 933–948, Dec. 1999. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--CooperatingJobs.ps

[27] F. Leymann, "Web Services Flow Language," Web page, May 2001. [Online]. Available: http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[28] Distributed Systems Department Pervasive Collaborative Computing Environment Project (PCCE), LBL, "PCCE Quarterly Reports," April 2002. [Online]. Available: http://www-itg.lbl.gov/Collaboratories/quarterly-reports.html

[29] S. Thatte, "XLANG: Web services for Business Process Design," 2001. [Online]. Available: http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

[30] F. Curbera, Y. Goland, Y. Klein, F. Leymann, D. Roller, and S. Weerawarana, "Business Process Execution Language for Web Services," Web page, version 1.0 - July 31, 2002. [Online]. Available: http://www.ibm.com/developerworks/library/ws-bpel/

[31] "Comparison of daml-s and bpel4ws (initial draft)," Sept 2002. [Online]. Available: http://www.ksl.stanford.edu/projects/DAML/Webservices/DAMLS-BPEL.html

[32] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic, and S. Zimek, "Web Services Choreography Interface," June 2002, version 1.0. [Online]. Available: http://wwws.sun.com/software/xml/developers/wsci/index.html

[33] A. Arkin, "Business Process Modelling Language," June 2002. [Online]. Available: http://www.bpmi.org/bmpi-downloads/BPML-SPEC-1.0.zip

[34] T. Haupt, E. Akarsu, and G. Fox, "Webflow: A framework for web based metacomputing," in *HPCN Europe*, 1999, pp. 291–299.

[35] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: a computation management agent for multi-institutional grids," in *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium.* San Francisco, CA, USA: IEEE Computer Society Press, Aug 2001, pp. 55–63. [Online]. Available: http://xplore2.ieee.org/iel5/7513/20446/00945176.pdf?isNumber=20446

[36] D. Gannon, R. Ananthakrishnan, S. Krishnan, M. Govindaraju, L. Ramakrishnan, and A. Slominski, "Grid Web Services and Application Factories," June 2002. [Online]. Available: http://www.extreme.indiana.edu/xcat/AppFactory.pdf

[37] "Condor : High Throughput Computing," 2002. [Online]. Available: http://www.cs.wisc.edu/condor/

[38] R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, and B. Smolinski, "Toward a Common Component Architecture for High-Performance Parallel Computing," in *Proceedings of High Performance Distributed Computing*, CCA Forum. Redondo Beach, California: , 1999, pp. 115–124. [Online]. Available: http://citeseer.nj.nec.com/context/1079818/225047

[39] R. Chinnici, M. Gudgin, J.-J. Moreau, and S. Weerawarana, "Web Services Description Language Version 1.2," July 2002, w3C Working Draft 9. [Online]. Available: http://www.w3.org/TR/2002/WD-wsdl12-20020709/

[40] The Hewlett-Packard Company, "Web Services Conversation Language (WSCL) 1.0," March 2002. [Online]. Available: http://www.w3.org/TR/wscl10/

[41] K. Amin, S. Nijsure, and G. Laszewski, "A Grid Services Framework for Collaboratorive Applications," in *SC'2002*, Baltimore, MD, 11-16 Nov. 2002, (Poster).

[42] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, and C. Kesselman, "Grid service specification," http://www.globus.org/reserach/papers/gsspec.pdf, February 2002. [Online]. Available: http://www.globus.org/reserach/papers/gsspec.pdf

[43] D. C. Marinescu, *Internet-Based Workflow Management: Toward a Semantic Web.* John Wiley & Sons, Inc., 2002.