12-2004

The Java CoG kit grid desktop : a simple and central approach to grid computing using the graphical desktop paradigm.

Pankaj R. Sahasrabudhe 1980-
University of Louisville

THE JAVA COG KIT GRID DESKTOP: A SIMPLE AND CENTRAL APPROACH TO GRID COMPUTING USING THE GRAPHICAL DESKTOP PARADIGM

By

Pankaj R. Sahasrabudhe
B.S., University of Louisville, 2003

A Thesis
Submitted to the Faculty of the
University of Louisville
Speed Scientific School
as Partial Fulfillment of the Requirements
for the Professional Degree

MASTER OF ENGINEERING

Department of Computer Engineering and Computer Science
University of Louisville

December 2004

The submitted manuscript has in part received funding from the Java CoG Kit, under the supervision of Dr. Gregor von Laszewski, and created by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under Contract No. W-31-109-ENG-38 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

An update to this thesis in the form of a technical report will be available at the end of January, 2005. The report will include more information about the Java CoG Kit and the Grid Desktop, and will be made available through http://www.cogkit.org or by contacting Gregor von Laszewski (gregor@mcs.anl.gov). Any work presented in this thesis should be referred through the technical report only.

THE JAVA COG KIT GRID DESKTOP:
A SIMPLE AND CENTRAL APPROACH TO GRID COMPUTING
USING THE GRAPHICAL DESKTOP PARADIGM

Submitted By: _____
Pankaj R. Sahasrabudhe
B.S., University of Louisville, 2003

A Thesis Approved on

6th December, 2004

by the Following Thesis Committee:

_____
Dr. Rammohan Ragade, CECS Dept., Thesis Director

_____
Dr. Gregor von Laszewski, Argonne National Laboratory

_____
Dr. Anup Kumar, CECS Dept.

_____
Dr. Larry Tyler, ME Dept.

_____
Dr. Eric Rouchka, CECS Dept.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

This thesis would not have been possible without the support and encouragement I received from several people. First of all, I am thankful to my advisor Dr. Rammohan Ragade for accepting me as his student and supporting my wish to pursue a thesis research appointment at the Argonne National Laboratory. His constant encouragement, persistence, and advice have become the source of my ambitions. He is truly a great advisor, professor, and an asset for our department. I am glad I heeded his advice.

Next, I want to thank Dr. Gregor von Laszewski who was my advisor at the Argonne National Laboratory. Words fall short to describe his contribution towards this thesis and also my personal development. I thank him for letting me pursue this project as part of the Java CoG kit group, and believing in my ability from the beginning. He has not only been a great mentor but a constant source of encouragement, strength and energy combined with an over achieving work ethic. Under his guidance, I have learned invaluabe skills which will help me throughout my career.

I would like to thank Dr. Anup Kumar, Dr. Larry Tyler and Dr. Eric Rouchka for being part of my committee. I acknowledge Dr. Kumar's guidance during initial discussions regarding thesis research topics. His advice was crucial for my decision to pursue a Grid Computing research. I appreciate all the efforts Dr. Tyler has taken to be a great mentor and a teacher. I am also grateful to Dr. Rouchka for guiding me with issues regarding bioinformatics.

Invaluable input was given by all the members of the Java CoG Kit group, especially Kaizar Amin and Mike Hategan. Having several discussions with them helped me better understand the Java CoG Kit and its role in Grid Computing. They also offered countless advice on many programming aspects of this thesis. I sincerely thank both of them for all

their efforts. I also thank all members of the Summer 2004 REU Program, namely, Robert Winch for his technical guidance, Matt Bone, Mike Sosonkin, and Nithya Vijayakumar for their help and support.

I cannot forget Mr. Ron Lile at the University of Louisville for his invaluable advice and guidance prior to my arrival at Argonne. He is an excellent mentor and a constant source of moral support. While working for him at the University, I have gained many skills that helped me successfully complete this thesis. I thank him deeply.

Also, I would like to acknowledge the emotional support I received from several of my friends both in Louisville, KY and Chicago, IL, notably Prachi Hote, Ashwin Cholpadi, Padmini Jayaraman, Rashmita Sahoo and Nanda Sreenivasan.

And last but most important of all, I would like to express my deepest love and respect for my parents, my brother, my sister-in-law and other family members for their constant encouragement, support and guidance. I want to thank my brother for introducing me to the world of Grid Computing and encouraging me to pursue a masters thesis. I thank my parents as they have always been there and believed in me in times when I did not. I feel I am blessed because they all are part of my life.

ABSTRACT

Grid Computing is evolving as a service based, flexible and secure resource sharing environment. Currently, with the help of Grid middleware toolkits, Grids are exposing their services through programming models and command line interfaces, requiring much technical knowledge of the backend Grid systems. Grid portals also exist, but fall short on integrating with native environments and maintaining a uniform user interface from portal to portal. In order to gain wider acceptance within the large and less technical oriented user communities, we need a homogeneous graphical user environment that supports the challenging task of providing Grid users an easy to use, seamless and transparent interface requiring minimal user participation.

Motivated by the needs of these users, we are presenting the Grid Desktop based on the popularity of the graphical desktop paradigms such as KDE and Windows XP. The Java CoG Kit Grid Desktop is a user centric workspace that enhances the normal operating system desktop paradigm by interlacing Grid concepts and leveraging commodity technologies like Java. The Grid Desktop contributes to the Java CoG Kit architecture and delivers ubiquitous computing through the Java CoG Kit abstractions, portability through XML and Java Web start technologies, and a simple user interface by following the vastly popular desktop patterns such as drag-n-drop.

NOMENCLATURE

Abbreviation	Term
API	Application Programming Interface
CoG Kit	Commodity Grid Toolkit
GGF	Global Grid Forum
GRAM	Grid Resource Allocation and Management
GT	Globus Toolkit
GT2	Globus Toolkit 2.0
GT3	Globus Toolkit 3.0
GT4	Globus Toolkit 4.0
HCI	Human Computer Interaction
HTTP	Hyper Text Transfer Protocol
IT	Information Technology
MDS	Monitoring and Discovering Service
OS	Operating System
RSL	Resource Specification Language
PSE	Problem Solving Environment
VO	Virtual Organization

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Grid Computing is a methodology to address extremely large compute intensive application needs. Although the infrastructure is being developed, most users are experiencing difficulty using the corresponding set ups due to unfamiliar and heterogeneous user interface environments. To address these issues, the Grid Desktop research initiative was organized by the Java CoG Kit group [7, 8] at Argonne National Laboratory [9], investigating a collaboration of three evolving concepts, namely: *Grid Computing*, *Problem Solving Environments (PSE)*, and *Human-Computer Interaction (HCI)*.

This section provides a brief introduction and a relation between the three main concepts. Later, an evaluation is presented based on those concepts, followed by the motivation behind this research project.

A. Grid Computing

Heraclites, an ancient Greek orator, is credited to saying that the only thing constant is change. Notably, throughout history a small set of these changes have lead to some large scale evolutions. For example, the invention of the steam engine in 1712 and later the telegraph in 1836 are considered to be key innovations that drove the industrial revolutions. More recently in 1948, the invention of the transistor opened the information age we live in today, replacing inefficient vacuum tube technologies and forever changing the way electronics impact our lives. Similarly, it is predicted that *Grid Computing* will have the same, if not greater, effect for this world and forever revolutionize every aspect of Information Technology [10] and its vast networked technologies like bioinformatics, finance, physics,

chemistry, and business, to just name a few.

So, what exactly is *Grid Computing*? What is its motivation? How is it being applied today? How will it shape our future? Before we address these questions, let us explore a scenario from the near future where *Grid Computing* technologies prevail.

Grid Scenario: John is about to leave for work. As he approaches his car in the morning, he feels nauseated and feverish. Looking at the watch showing 9:05 am, he realizes that he cannot fall sick and miss his important business appointment in the morning. Once in the car he quickly pulls out his cell phone, places his finger on the Red Cross button, and waits for his medical diagnosis. The button is equipped with a blood sampling probe which extracts a sample of his blood and sends the information to a medical facility nearby. Based on the sample, the computer system at the facility starts diagnosing his blood while retrieving his medical history from his physicians office. At 9:06 am, a notification is sent to his cell phone stating his sickness, the possible causes, and that his medication is being prepared. In the meantime, a computation is started based on his diagnosis and medical records to design a customized drug, which within minutes is sent to his pharmacy. On his way to work John stops by his pharmacy to pick up his drug and sign the credit card payment receipt.

Does this sound hard to believe? The scenario might sound far fetched at present, but emerging *Grid Computing* technologies hold a promising drive to satisfy every aspect of this scenario.

Grid Computing is defined as an infrastructure that allows for flexible, secure, coordinated resource sharing among dynamic collections of individuals, resources, and organizations [11]. Parallel to the above scenario, the cell phone, computer systems, and the pharmacy create a virtual organization that securely exchanges medical information and provides the user with a complete and reliable medical service. The user was not aware of how, when, and what happened in order to obtain his diagnosis and the customized drug

to cure his sickness. While in the background, massive computing power was consumed to quickly formulate the structure of his drug based on his current diagnosis and medical history; a data link was established between the nearby medical facility and his physicians' office in order to transfer personal medical files; the drug manufacturing specifications were transferred to a reliable pharmacy before the person arrived to collect it. All these transactions had to be secure, reliable, distributed, involving a heterogeneous set of resources while preserving information integrity. Any loss or tampering of drug information could result in deadly symptoms.

It is clear that without an infrastructure, distributed tasks would be impossible to coordinate. As pointed out in [12], "An infrastructure is a technology that we can take for granted when performing our activities. The road system enables us to travel by car; the international banking system allows us to transfer funds across borders; and the Internet allows us to communicate with virtually any electronic device." Luckily, *Grid Computing* technology infrastructure is being developed under the auspices of the Global Grid Forum (GGF), which is an international community-initiated forum of individual researchers and practitioners working on various facets of Grids [2]. The GGF oversees all Grid related activities including applications, development, and community relations (see Figure 6). Industry leaders like IBM, Sun and Intel are joining hands with the major research centers and academic institutions to standardize protocols and services used by grid resources.

Knowing the distributed nature of *Grid Computing* and its focus on sharing resources among organizations, we introduce the notion of a Virtual Organization (VO). As noted in [11] the resource sharing, necessarily, is "highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs" A set of individuals and/or institutions defined by such sharing rules form what we call a Virtual Organization (VO) (see Figure 1).

Reflecting back to the *Grid Scenario*, a VO was formed consisting of the pharmacy,

patient and the hospital. Each operated only in their allowed authentication modes. For example, the pharmacy could not access the patient data directly, it was handed the information by a delegated authority like a hospital.

The enabling factor for VO's is the underlying Grid architecture consisting of five layers [13]. Each layer builds on top of the other, creating a chain of interoperability between heterogeneous resources. They are similar to the layers in computer architecture which progress from hardware layer to application layer. The following is a summary of Grid layers largely based on the text explained in [11].

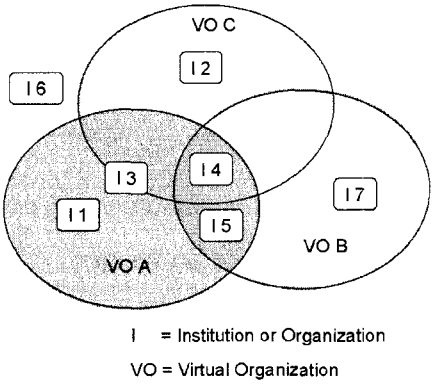

Figure 1: Virtual Organizations [1].

Fabric This layer provides the interface to the local hardware including computational resources, storage systems, catalogs, network and sensors, making it the most basic layer. As resources get more and more distributed this layer plays a key role in maintaining standard protocols to access heterogeneous resources.

Connectivity Defines core communication and authentication protocols required for Grid-specific network transactions to and from the Fabric layer. Communication requirements include transport, routing, and naming and are mostly based on the TCP/IP protocol. The authentication protocols build on communication services to provide

cryptographically secure mechanisms for verifying the identity of users and resources and include the following features (see Figure 2).

Single sign on Users must be able to "log on" to multiple resources once without needing user intervention.

Delegation A user must be able to grant access to a program on that user's behalf, so the program can access authorized resources according to the users authentication.

Integration with various local security solutions Local security infrastructure must be compatible with Grid security solutions. Realistically, the Grid security solutions should not force to replace existing local security solutions, but instead provide the mapping required.

User-based trust relationships If a user is authenticated to use multiple resources, that user should not have to deal with security administrators from those multiple sites every time the sites are accessed.

Resource This layer builds on the connectivity layer communication and authentication protocols to define protocols for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources. Resource layer is only concerned with individual resources and ignores issues regarding distributed management of resources as this is handled by the Collective layer.

Collective Collective layer is responsible to manage multiple distributed Resource layers, offering co-allocation, directory, monitoring, data replication, software discovery, scheduling and brokering services implementing a wide variety of sharing behaviors without placing new requirements on the resources being shared.

Application This is the final layer that consists of user applications used in the VO, which in turn could use sophisticated software development kits like the Java CoG Kit [7].

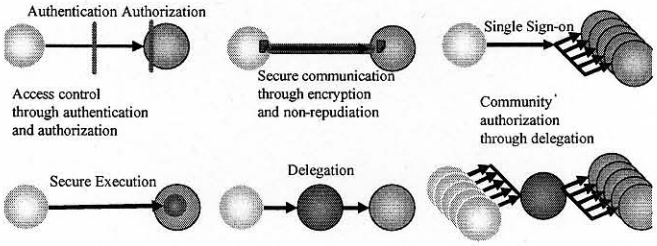

Figure 2: Simplified Grid Security Concepts [2].

In essence, the Grid architecture combines security management with general Grid management entities like tasks, data, communities and services (see Figure 3). A good summary of these issues are explained in [2]. Essentially, Grid management oversees interactions between socio-political entities and the Information Technology (IT) infrastructure. Analogous to the *Grid Scenario*, this reflects managing which medical data can be shared among the pharmacy, hospital, physicians' office, and the patient, which usually transitions from less access to more, as the patient has the rights to view all his medical data, and the pharmacy only deals with data it is presented.

At the core of Grid management lies security issues like authentication, authorization, encryption, and nonrepudiation. According to [2], which states:

Authentication is the verification of the identity of an entity within the Grid. Though this is commonly associated only with identification of a Grid user, the Grid also requires authentication of resources and services provided as part of the Grid.

Authorization deals with the verification of an action an entity can perform after authentication was successfully performed. Thus, policies must be established to determine the capabilities of allowed actions. A typical example is of a batch queue that allows user A to use a resource between 3 and 4 o'clock, but user B to use between 5 to 6 o'clock. In general, policies determine who can do what, when, and at which resource.

Encryption provides a mechanism for protecting the confidentiality of messages in transit between two peers.

Nonrepudiation prohibits resource gateways to arbitrarily deny that it granted a resource reservation.

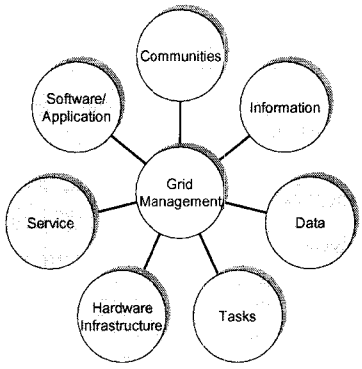

Figure 3: Grid Mangement [2].

These security issues extend security concepts like Single sign on, Delegation, Community authorization and Secure Execution of the Collective Grid architecture layer, completing the circle of grid security requirements (see Figures 2 and 4).

B. An Evaluation

The previous section introduced the basic concepts of *Grid Computing*. In this section we evaluate where *Grid Computing* stands today and examine room for possible improvements. Later, we conclude this section with factors motivating this thesis project. To begin with, we note that Grids can be classified into two parts. One part that provides func-

tionality supported through its association with Problem Solving Environment (PSE), and the other part providing usability through association with Human Computer Interaction (HCI) (see Figure 5). In the following section we relate these two research fields to *Grid Computing*, starting first with PSE.

Everyday eBay [14] is enabling sellers with surplus resources to connect with the potential buyers having demand for those resources. The eBay web-portal is essentially solving community problems by providing a link between the resource voltage, in turn classifying eBay as a user centric PSE. Although the field of PSE is relatively new and difficult to summarize, the pioneers defined it as "a computer system that provides all the computational facilities necessary to solve a target class of problems", and further PSEs "use the language of the target class of problems, so users can run them without specialized knowledge of the

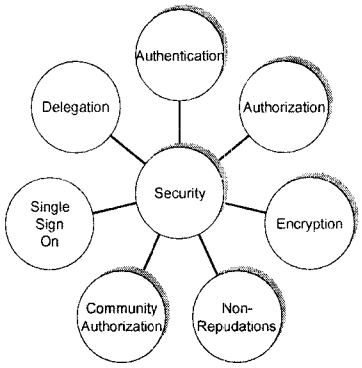

Figure 4: Grid Security [2].

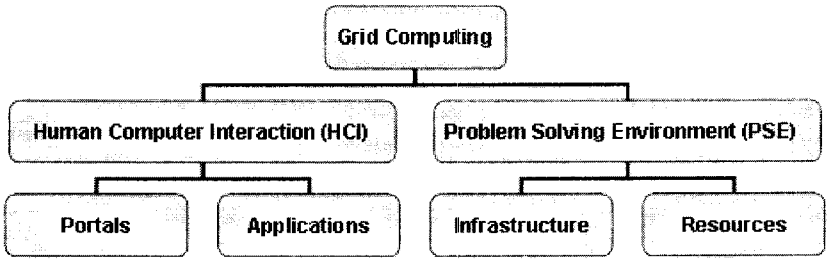

Figure 5: Thesis Project Integrals

underlying computer hardware or software" [15]. Essentially, a PSE serves as a mediator between the user and the problem, providing all necessary tools to solve a problem efficiently without distracting the user with unnecessary complexities [16]. A few examples of existing PSEs are Grid Computing through spread sheets [17], Netsolve - distributed mathematical solver [18, 19] and Grid enabled PSE [20]. Even cluster computing infrastructures like Beowulf [21] and Mosix [22] can loosely be considered a PSE, as they provide a computational problem solving sandbox. Similarly, *Grid Computing* can be considered a multi faceted PSE as it connects resources and communities [23] and solve a variety of problems defined by the participating VOs. A Grid PSE even has the ability to combine several heterogeneous PSEs into a single collaborative PSE. But *Grid Computing*, as implemented today, cannot fully be considered as a PSE as there is no single homogeneous environment that provides all problem solving tools. This research project attempts to tackle this issue by providing a user centric PSE providing all necessary tools to interact with Grids and also enhance use experience by following several aspects of HCI, as described below.

HCI [24] is a research field concentrated on creating a system that is easy to learn and easy to use [25]. Through development of new paradigms as mentioned in [26, 27], the systems comply with common HCI goals. For example, several desktop paradigms have been developed to support the underlying system with an easy to use user interface [28, 29, 30, 31]. Are Grids today taking similar efforts to support the HCI goals? If not, what can we do to achieve these goals? Currently, HCI goals are being addressed but there is great room for improvement. For instance, in order to complete a task, the average Grid user has to interact with multiple user interface environments like command-line interfaces and programming models, requiring technical knowledge of the backend Grid systems. Grid portals exist, but fall short on providing the same user interface from portal to portal. The portals also have difficulty integrating with native OS patterns like drag-and-drop, which boost usability. In order to gain wider acceptance within the large and less technical

user communities, we need a homogeneous graphical user environment that supports the challenging task of providing Grid users an easy to use, seamless and transparent interface requiring minimal user participation. This is the core motivation for this project, and an overview is presented in the next section.

C. Overview of Project

The goal of this project is to address common HCI goals [32] along with PSE requirements in providing a functional user interface that provides seamless Grid functionality. We introduce the desktop paradigm called "Java CoG Kit Grid Desktop", similar to Windows and KDE, and that is easy to learn, easy to use, and integrates Grid activities into existing Operating System (OS) environments. The Grid Desktop delivers ubiquitous computing through the Java CoG Kit abstractions, portability through XML and Java Web start technologies, and a simple user interface by following the vastly popular desktop patterns such as drag-and-drop. Because "Without users there can be no Grid" [33], we must provide users the usability they need to fully exercise Grid functionality.

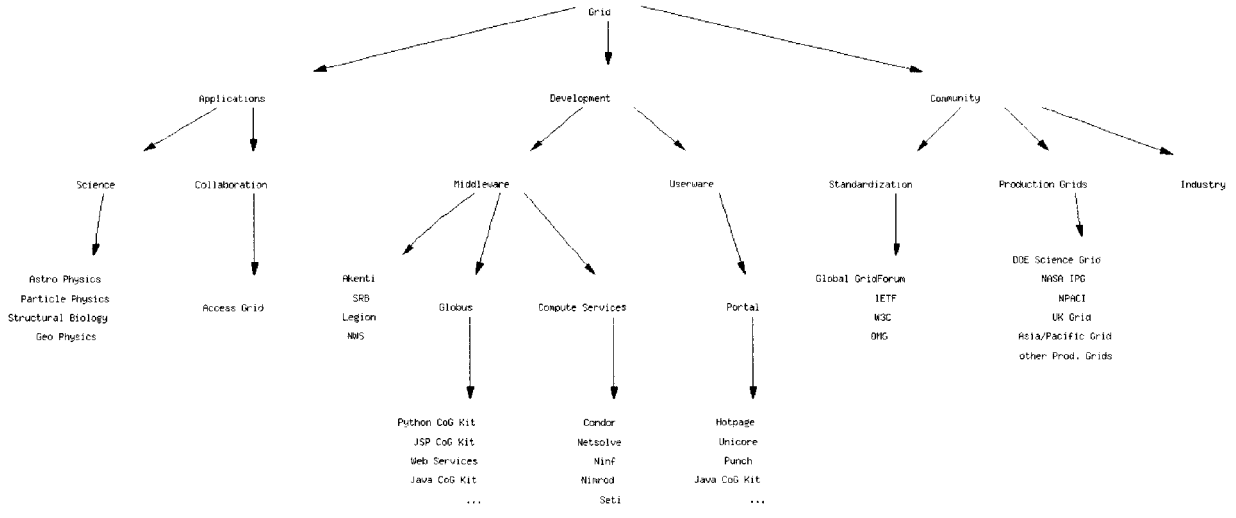

Figure 6: A Simple Classification Of Grid Activities: community activities, development tools and applications [2].

II. RELATED WORK

A. Overview

This section provides an overview of existing work that either resembles or complements the Java CoG Kit Grid Desktop and its components. As noted in the previous section, *Grid Computing* encompasses several aspects of Problem Solving Environments (PSE) and Human Computer Interaction (HCI). This creates a vast structure of contributing communities and organizations in an effort to coordinate resources using standard, open, general-purpose protocols and interfaces delivering nontrivial qualities of service [34]. Several sources [35, 36] are dedicated to inform Grid communities with news and information about latest Grid developments. Thus, covering all related concepts would not be necessary and we shall only explore concepts directly related to the Java CoG Kit Grid Desktop.

The Java CoG Kit Grid Desktop is a desktop paradigm similar to KDE, MAC OS and Windows that consist of a toolbar, icons, frames, and menus supporting common desktop patterns like drag-n-drop independent of the underlying OS. To be able to utilize this paradigm in the context of *Grid Computing*, we rely on underlying middleware and workflow components, which in turn execute the application tasks (see Figure 7).

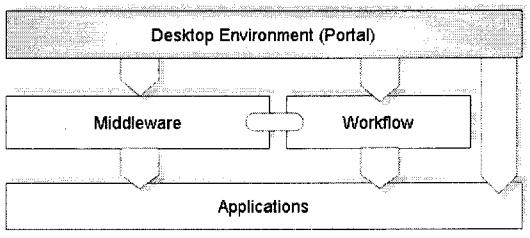

Figure 7: Collocation Of Grid Desktop Related Components

In the following sections we explore relevant middleware, workflows, and portals, followed by a feature comparison to summarize the key points presented in this section.

B. Grid Middleware

As defined in [1] middleware consists of protocols, data structures, and objects that can be accessed through convenient APIs and classes (see Figure 8). Essentially they bond the applications to the underlying Grid complexities [37] (see Figure 9), proving their effectiveness through smaller projects like eMinerals [38] to larger projects like the Terra Grid. Middleware technologies have increasingly become a service based architecture by accepting the Open Grid Services Architecture (OGSA) [39] providing elementary Grid services like job execution services, information services, file transfer services, and security services.

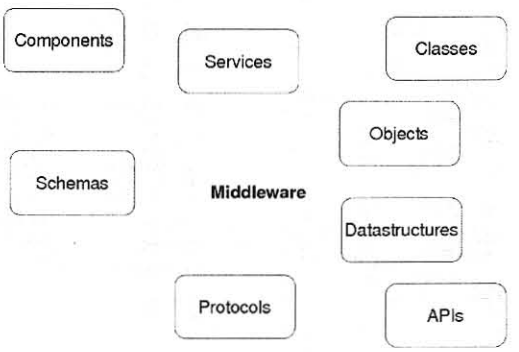

Figure 8: Grid Middleware Overview [1]

The elementary Grid services are mostly used during synchronous Grid interactions. To suffice the asynchronous nature of Grids, the middlewares are striving to develop advanced Grid services such as File Management, Task Management, and Information Management services. File Management services provide the ability to dynamically adapt to changing network conditions, and fault situations, taking care of transfers for the users without their intervention. Task Management services help users manage large number of tasks, which

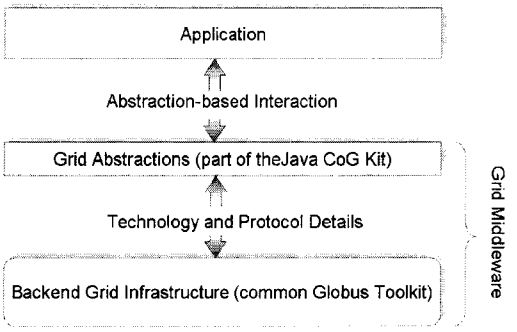

Figure 9: Grid Middleware Layers [3]

entitles management of distributed elementary Grid services (see Figure 10). And Information Management Services manage and monitor Grid resources supporting administrative efforts.

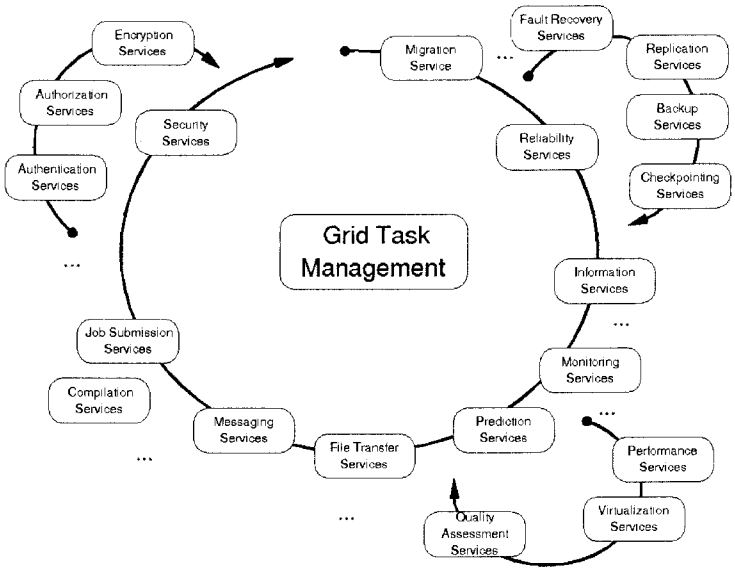

Figure 10: Grid Task Management [1]

Middleware is an important component of the Grid Desktop as it ties the Grid functionality to the user interface. In the following sections we shall examine few of these relevant middlewares.

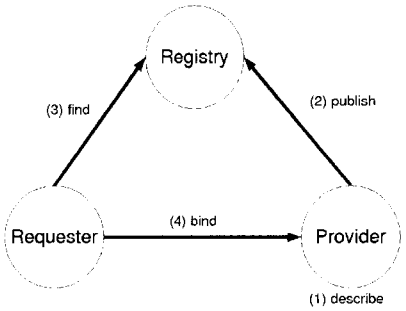

Figure 11: Grid Services [1]

B.1 The Globus Project

The Globus project [40] has contributed in several ways to the Grid community. Essentially all major Grid projects are being built on protocols and services provided by it [41]. We can classify these contributions into five major areas [1]. First, conducting research on Grid-related issues such as resource management, security, information services, data management, and application development environments. Second, development of the open source Globus Toolkit making it the de-facto industry implementation of the Grid protocols. Third, assisting and building large-scale testbeds for scientists and engineers. Fourth, collaboration with large number of application oriented efforts that develop large-scale Grid-enabled applications with scientists and engineers. And finally, contribution through community activities that include educational outreach and participation in the Global Grid Forum to define Grid standards.

The Globus Toolkit started by providing just APIs, but has evolved to provide protocols and services like Metacomputing Directory Service (MDS) [42], Grid Resource Allocation and Management (GRAM) [43], and their integration with commodity toolkits like the Java CoG Kit [44]. The toolkit can be divided into Security, Communication, Information, Resource Management, Data Management and Data Grid features, as explained below.

Security in Globus Toolkit is built around the Grid Security Infrastructure (GSI) which

uses public key cryptography as the basis for its functionality. It enables key security services such as mutual authentication, confidential communication, delegation, and single sign-on [1]. Other services rely on this infrastructure for secure distributed communication. Globus also provides portals to acquire the certificates used to authorize Grid access [45].

The GlobusIO API handles the communication within the toolkit. Services like TCP, UDP, IP multicast, and file I/O with support for security, asynchronous communication, and quality of service are provided [1]. A Grid File Transfer Protocol [46] is also presented to perform file transfers over Grid resources. Through the MPICH-G2 [47], support for MPI across distributed resources is also acquired.

Information about the Grid is handled through the MDS using Lightweight Directory Access Protocol (LDAP). Resource Management is delegated to the local resource allocation services [1]. Data Management is supported by integrating GSI protocols with existing HTTP and FTP services. Data Grids are provided through replica catalog services by allowing copying of the most relevant portions of a data set to local storage for faster access [1].

Many community applications are extending the Globus Toolkit to enhance and optimize the middleware to suit their needs. The IBM Grid initiative is one such example [48] that provides integration with Websphere and other e-Business technologies.

B.2 Legion

Legion was developed at the University of Virginia, and is now part of the Avaki corporation [49]. A comparison between Legion and the previously mentioned Globus is presented in [50]. The goal of Legion is to support parallelism in application code while managing the complexities of the physical systems for the user, and schedule distributed processes on available and appropriate resources while providing the illusion of working on a single, virtual machine [1]. Legion provides advanced services like automatic installation of

binaries, secure and shared virtual file system that spans all the machines in a Legion system, strong PKI-based authentication, flexible access control for user objects, and support of legacy codes execution and their use in parameter space studies [1]. Much of Legion's architecture is based on an object model where each entity in the Grid is represented as an active object that responds to member function invocations from other objects. To support this framework, Legion provides several core objects, such as compute resources, persistent storage, binding objects that map global to local process IDs, and implementation objects that allow the execution of machine code [1]. Legion follows standard message format and high-level protocols for these object interactions, but does not restrict the programming language or the communications protocol. Users can extend core Legion objects to define their own objects, making it scalable and flexible.

B.3 Condor

Condor [51] is a specialized job and resource management system for compute intensive jobs that provides job management mechanism, scheduling policy, priority scheme, resource monitoring, and resource management [52]. Condor maintains a pool of computers while using a centralized broker to distribute jobs based on load information or through preferences set by the jobs to be executed. The proper resources are found through the ClassAds mechanism of Condor that allows each computer in the pool to advertise the resources which it controls and publish them in a central information service. Any workstation in the Condor pool may be called by the broker to compute, and thus Condor has the ability to delegate interactive use of the workstation to the console users. Check pointing and migration of jobs to other host machines is also supported. Condor is also making efforts to comply with other middleware's like Globus [52] to enable larger collections of resources that span across multiple domains.

B.4 myGrid

myGrid is a middleware specifically designed for bioinformatitians. It supports resource discovery, workflow enactment and distributed query processing [53]. Currently, myGrid is being developed to conform to the OGSA [39] standards and support a service based grid architecture that can be divided into three categories: services that perform experiments, services that discover and manage metadata, and services which collaborate with other e-Science initiatives. First of these services provide necessary access to bioinformatics tools like NCBI BLAST [54], WU BLAST [55], and the complete EMBOSS [56] application suite containing several analysis packages. Workflow Enactment and distributed database queries are also included through these services. The discovery and management services provide interface to the UDDI [57] that registers services together with metadata about their location, ownership, version, cost, quality of service, and security [53]. The collaborative e-Science offers services dealing with notification, personalization and provenance [53]. An example myGrid application can be noted in [58].

B.5 The Java CoG Kit

Currently, the Globus project provides elementary Grid middleware services without extensive support for commodity technologies used by Grid developers. Commodity programming environments like [59] try to resolve this issue by providing tools and APIs but do not solve the problem completely. In a mission to irradicate this issue, the Commodity Grid project [60] is creating Commodity Grid Toolkits (CoG Kits) that define mappings and interfaces between Grid services and the particular commodity frameworks of interest, including Java, Python, CORBA, Perl, Web Services, .NET, and JXTA [61]. The Java CoG Kit is one such implementation which reimplements the Globus protocols in Java and provides advanced services currently not available in the Globus Toolkit. The Java CoG Kit is also a framework for designing computing portals [1] and has proven itself to become an in-

tegral part of GT3 along with several projects at major organizations like CERN, DOE and NSF. Notably, it also won the "Best Poster Award" at the 2004 Super Computing Conference [62]. Because the Java CoG Kit holds these features, it best serves as the middleware of choice for this project. An overview of the Java CoG Kit is depicted in Figure 12, and a comprehensive manual is available online [4].

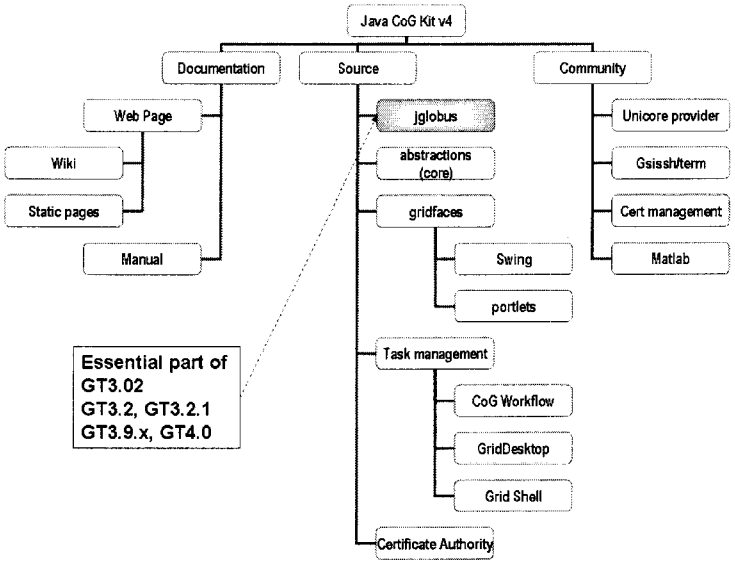

Figure 12: Java CoG Kit 4 Overview [4]

C. Workflows

Workflows are the operational aspects of a work procedure containing a collection of tasks with or without dependencies. Figure 13 depicts a good overview of how workflow complexity increases as systems transition from a single user to Grid communities. Several workflow tools exist that can create, monitor, and manage individual tasks [63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73]. A complete online survey providing up to date information pertaining to workflow activity can be found at the Indiana University Extreme labs site [74]. Relative to the Java CoG Kit Karajan workflow tool, these workflows provide just the basic functionalities. The Karajan workflow (successor of GridAnt [5]) provides extensive

support by defining a combination of distributed parallel and sequential tasks, supporting the Grid security features and services, and providing a sophisticated visual workflow editor. The Grid Desktop benefits from the Java CoG Kit framework and its Karajan workflow system as they provide a well rounded package of functionality and usability, satisfying PSE and HCI goals.

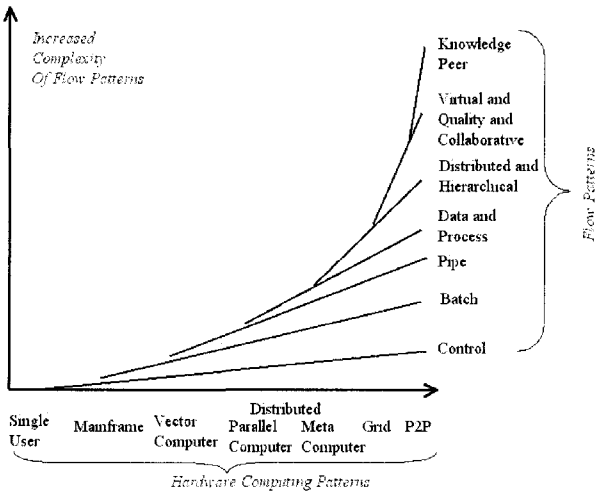

Figure 13: Flow History [5]

D. Portals

The Java CoG Kit Grid desktop is a portal. A Portal being a community service with a single point of entry to an integrated system providing access to information, data, applications, and services [75], addressing common HCI and PSE issues mentioned earlier. The term Portal can be refined depending on the environment. For example a Web Portal accesses information through Web browsers using several Web-based commodity technologies like HTTP and CGI. Similarly, Grid portals, like the Grid Desktop, can be defined as a specialized portal to access information, data, services and applications residing on Grid resources. Grid portals may contain heterogeneous types of portals in an effort to address advanced HCI issues. Several attempts are being made to integrate Grid portal concepts

into existing user desktop environments and providing users instant access to Grids without additional knowledge. These attempts and several other relevant Grid portals are presented in this section.

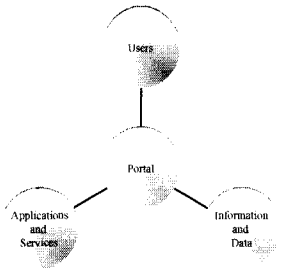

Figure 14: Portal [2]

Grid portals may have varying user interfaces but all have proven to be simple and effective Grid environments [76]. They mostly vary as they are created using different portal generators like Web portal generation toolkits similar to JetSpeed [77]. Other portal toolkits like Portal Construction Toolkit for the Grid (PCT4G) [78], Legion Grid portal [79], Jini-based Portal Augmenting Grids (JiPANG) [80] and Grid Speed [81] provide pre-packaged Grid portal creation frameworks for faster time to market. Several specilized portals also exist. For example, in the bioinformatics domain we have Bio Portal [82], Proteome Annotation Portal [83], Bioinformatics portal [84], BioBar [85], Helmholtz Network [86] and Hembase portal [87] providing a mixture of information and Grid access portals. In addition several organizations, like The Open Grid Computing Environment (OGCE) [88] assist Grid projects in providing user interfaces and environments, including portals.

Let us briefly review a few popular Grid portals. The UNICORE (UNiform Interface to COmputing REsources) [89] portal provides a simple access to the Grid including workflow management, job submission, job monitoring, and job control as part of a single client. Similarly, HotPage [90] is a portal that provides a collective view of a distributed set of

computing resources while providing simple resource management and query services. Additionally, HotPage supports the Globus services and enables users to access and manipulate files and data to submit, monitor, and delete jobs. Also, the Talisman [91] portal has successfully provided portal interfaces for the myGrid bioinformatics middleware mentioned earlier, supporting workflow and information queries for the Grid.

Aside from providing Web based portals, several stand alone application portals are also available. One of which is the Java based Entrada Project [92] which is a light-weight application hosting framework geared to help make Grid resources easier to access and use. Several similar tools exist like the CoG Box [93] and Generic GUI Generator for the Grid (GuiGen) [94]. Also, applications like the @Home project [95, 96, 97] provide simple portals accessing compute cycles on public Grids through simple "screen saver" type executables.

D.1 Desktop Environments

All portals mentioned earlier are accessed through external applications, like Web browsers and application launchers, forcing users to use environments not directly associated with the OS. This may not always provide a homogeneous interface and additionally require the user to learn new skills complying with changing environments. We can solve this issue by integrating Grid functionality within the desktop paradigm itself. We have several options, one being to extend existing desktop environments like Windows [28], Mac OS, and Sun Desktop [29]. But they provide proprietary implementations of the desktop paradigms, and make it harder to extend and freely support Grid enabled functionalities. Alternatively, we could extend publicly available desktop technologies like KDE [98] and Zesktop [99] but would restrict users to use a specific OS. One of these attempts can be found through the Grenade project [100], which attempts to integrate job submission service into the KDE desktop.

There is need for a portable desktop paradigm that retains its user interface independent of the underlying OS. The Migrating Desktop [101] attempts to provide this implementation, but is specific to the Cross Grid and would require overhead to comply with other Grid infrastructures. The Java CoG Kit Grid desktop provides the optimal solution to this problem by building on the Java CoG Kit abstractions, which supports multiple Grid infrastructures, and by creating a portable implementation of the desktop paradigm that essentially creates a virtual OS interface conforming to a uniform look and feel. We shall now compare these options in the next section.

D.2 Feature Comparison

In this section we relate Grid portal technologies. As depicted in Figure 15 which compares usability (y-axis), functionality (x-axis) and availability (oval size) of several Grid portal classes. The usability encompasses all user driven aspects of the interface, addressing questions like does the portal have support for drag-n-drop, icons, frames , and other HCI aspects. The portal functionality quantifies support for various Grid systems like Globus, Condor, UNICORE, etc. And availability quantifies the approximate number of unique implementations per portal class.

In summary, although the Java CoG Kit Grid Desktop is far from replicating the well established proprietary desktop systems, it does provide a convincing solution for average Grid users who need Grid functionality quickly without learning new paradigms. The Java CoG Kit Grid Desktop is the first such implementation to leverage the desktop paradigm and apply it towards *Grid Computing*. Other projects like the Grenade project and the Migrating Desktop, emerged later in an attempt to provide a desktop paradigm on specific systems. The Grenade project currently only supports basic Grid services on KDE desktops and plans to support others like Redhat, Debian, and Windows. Similarly, the Migrating Desktop is designed to use with the Cross Grid infrastructure, and would require overhead

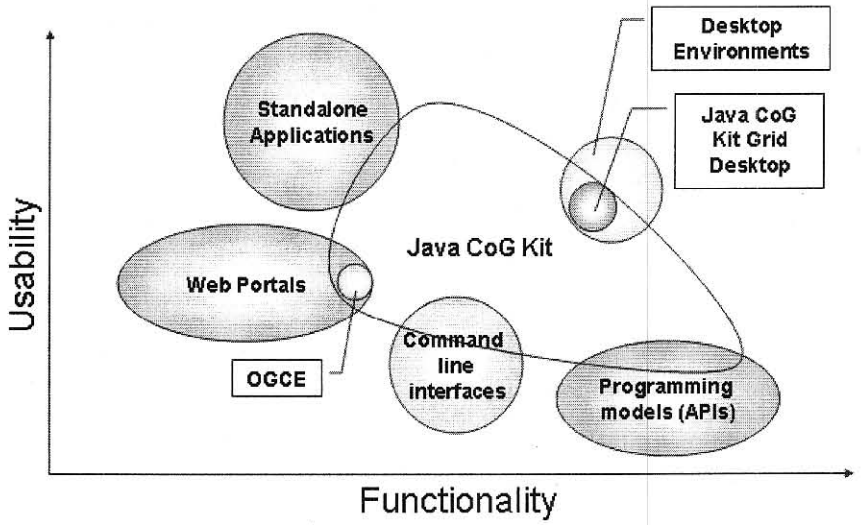

Figure 15: Portal Comparison [6]

to comply with a wide range of Grid systems. None of them support the wide range of underlying Grid architectures like the Java CoG Kit Grid Desktop. A summary is provided in Table D.2.

Table I: Desktop Paradigm Comparison

Name	Type	Released	User base	Usability	Grid functionality	Single OS dependent
Windows	Proprietary	Yes	High	High	None	Yes
Sun Desktop	Proprietary	Yes	Moderate	High	None	No
Mac OS	Proprietary	Yes	High	Very High	None	Yes
CDE	Proprietary	Yes	High	High	None	Yes
Zesktop	Proprietary	No	Low	Moderate	None	No
KDE	Open Src	Yes	High	High	Low	No
Clemantis	Open Src	Yes	Low	Low	None	No
Jdistro	Open Src	Yes	Moderate	Moderate	None	No
Jesktop	Open Src	Yes	Low	High Low	None	No
Grenade Desktop	Academic/Open Src	No	Low	Low	Moderate	Yes
Migrating Desktop	Academic/Open Src	Yes	Low	Low	Low	No
Java CoG Kit Grid Desktop	Academic/Open Src	Yes	Low	Moderate	High	No

III. THE JAVA COG KIT GRID DESKTOP

In the previous sections we established the basics of *Grid Computing*, motivated the need for a Grid Desktop, and explored related technologies. In this section we present an overview of the Grid Desktop architecture.

A. Overview

The Grid Desktop is part of the Java CoG Kit's *Gridfaces* module, which serves as an abstract interface connecting underlying Grid services with the applications (see Figure 16). The *Gridfaces* module contains several utilities and GUI applications like the Grid Shell and the Grid Directory Browser, which are accessible from the Grid Desktop and as stand alone applications. Additionaly, using the Java Beans technology, the Java CoG Kit plans to support a Grid Integrated Development Environment (IDE) for faster application development providing out of the box Grid enabled components.

Leveraging Java technologies and the Java CoG Kit, the Grid Desktop provides a well rounded user workspace that is easy to use and dynamic, containing basic desktop components like menubars, toolbars and icons. In addition the Grid Desktop icons support drag-n-drop and active state notifications that follow the state graph depicted in Figure 17.

Please note that the following sections are part of the Java CoG Kit 4 manual [4].

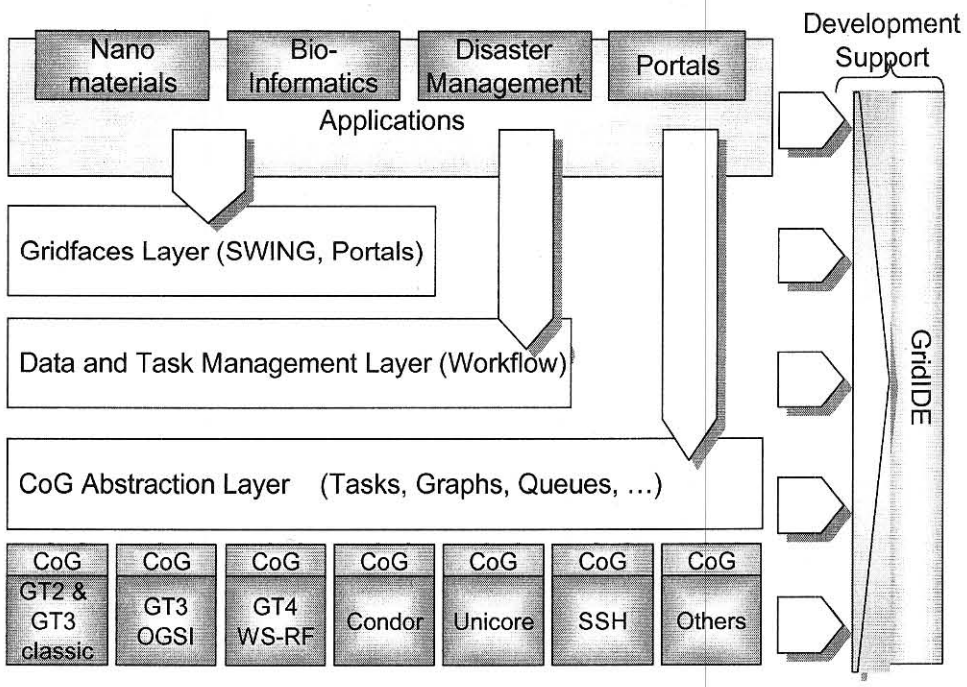

Figure 16: Architecture of the Java CoG Kit [4]

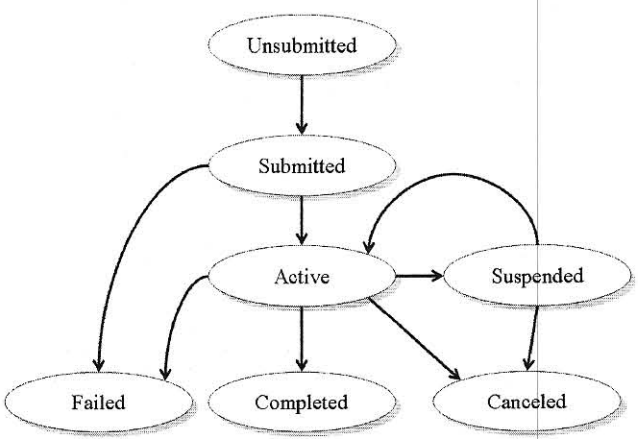

Figure 17: Active Icon State Graph [3]

B. Design

In this section we present the abstract desktop architecture for the Java CoG Kit Grid Desktop. It resides in the Java CoG Kit *Gridface* package and extends the much general *Desktop* interface from the *Gridface/interfaces* package. Together they form the package hierarchy depicted in Figure 18.

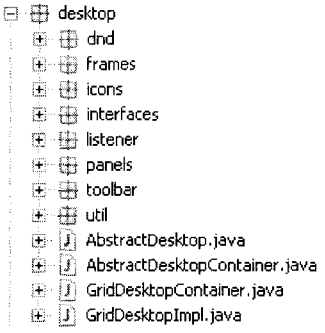

Figure 18: Desktop Package

The following summarizes the *desktop* package, linking relevant sub-packages to later sections.

dnd package provides the drag-n-drop support for the desktop icons (Section B.1);

frames provides the abstractions for creating desktop internal frames (Section B.2);

icons provides the desktop icon abstractions and its mouse listener (Section B.3);

interfaces provides the basis-set interfaces used by the desktop (Section B.4);

listener contains the mouse listener class for the desktop;

panels contains the Dynamic Form Panel (DFP) framework (Section B.5);

toolbar contains implementation for the desktop toolbar (Section B.6); and

util contains utility and supporting classes used by the desktop (Section B.7).

The *desktop* package contains two abstract classes along with their Grid enabled extending classes. One of the abstractions, the *AbstractDesktopContainer*, extends the *JFrame* class from Java and contains the menu bar, *DesktopToolBarImpl* and the desktop itself. The second abstraction, *AbstractDesktop*, extends the *JDesktopPane* class from Java and provides the abstract implementation for basic actions like adding and removing icons, and saving desktop state to a XML file. An overview of this architecture is presented in Figure 19.

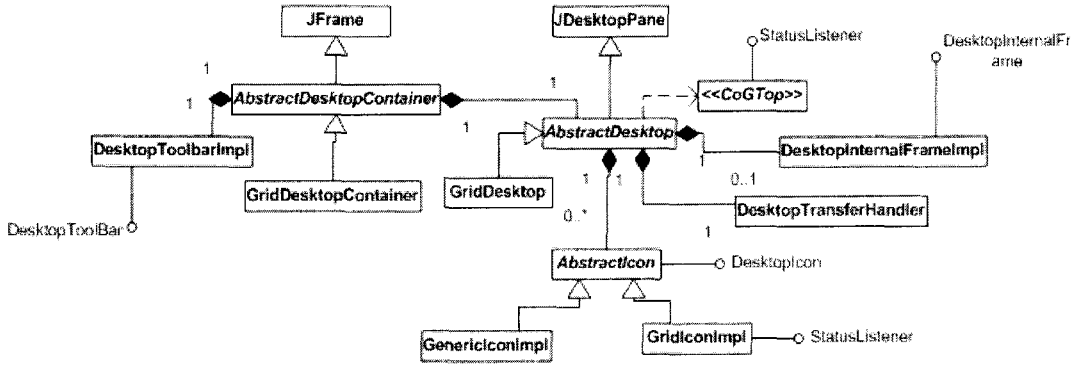

Figure 19: Simplified Desktop UML Diagram

B.1 Drag and Drop

Modern desktop paradigms have introduced the ability to drag and drop elements onto each other, provoking logical behaviors. For example, dragging and dropping a file on an executable might provoke the executable to open the dropped file. The Grid Desktop applies similar behaviors in the context of *Grid Computing* through two classes contained in the *dnd* sub-package.

The *DesktopTransferHandler* class extends the *TransferHandler* class from Java. It is the main class for handling drag and drop between icons, making full use of the Action Proxy Framework (APF)(see Section B.4). The *DesktopTransferHandler* divides the drag and drop actions into three stages. First stage prepares the drag icons by wrapping them

in a *DesktopIconTransferable* object having a *DesktopIconGroup* class flavor. The *DesktopIconGroup* class is contained in the *icons* sub-package (see Section B.3), and is the only flavor currently supported for inter desktop drag and drop. Once the transfer object is created the *DesktopTransferHandler* checks to see if the dragged over object supports the drop by delegating to the underlying APF. If the APF approves the drop then the *DesktopTransferHandler* again delegates the import drop action to the APF. In essence, by leveraging the APF the *DesktopTransferHandler* class serves as an abstract drag and drop handler for current Grid Desktop implementation and its future extensions.

The *DesktopTransferHandler* class also supports drag and drop between native OS icons and the Grid Desktop. Currently only two native icon flavors are supported, one is the *List* flavor used by Linux systems, and the other being *String* flavor used by Microsoft Windows systems. These native OS icons are added to the desktop by simply dragging them to the Grid Desktop.

B.2 Internal Frames

The Grid Desktop supports internal frames through the *DesktopInternalFrameImpl* class which extends the *JInternalFrame* class from Java. The *DesktopInternalFrameImpl* class implements the *DesktopInternalFrame* interface which specifies default desktop internal frame attributes and constants. The desktop internal frame content can be any object that extends the *Component* class from Java. In essence, supporting a wide variety of visual components and containers already existing in Java like *JTable* and *JPanel*. The *JPanel* container is used more often as it is light-weight and very flexible. Additionally, the *DesktopInternalFrameListener* class contained inside the *frames/listener* package, provides standard implementation for internal frame events like closing and opening. A couple of specialized internal frames are implemented for the Grid Desktop and are mentioned in Sections C.6 and C.7.

B.3 Icons

Desktop icons are the most basic element for user interaction within the Grid Desktop. They provide the ability to invoke internal frames, launch native OS applications, and support drag and drop between other icons. Much of the abstract icon behavior is captured in the *AbstractIcon* class, which also provide access to the basic icon attributes like icon text and image location. These attributes can be edited through the Properties action of the icon configurable popup.

Each desktop icon is associated with an icon type. To begin with, the abstract desktop architecture provides the NATIVE and SYSTEM icon type. The NATIVE icon type is assigned to icons that invoke native OS applications from the icon, and the SYSTEM icon type is reserved for preconfigured un-editable icons. More icon types can also be added, as seen in the next section which adds Grid enabled icon types. Both the NATIVE and the SYSTEM icon types are implemented in the *GenericIconImpl* class, which extends the *AbtractIcon* class. All basic icon attributes can be edited through the icon Properties popup action, which opens a Properties dialog frame for all except the un-editable SYSTEM icons.

The NATIVE icon Properties dialog frame is organized into two sections, bottom section for "Basic Icon Properties" (from *AbstractIcon*) and the top section for "Generic Icon Properties" (from *GenericIconImpl*). As NATIVE icons can execute native OS executables, there is a text field to specify executable name and its arguments as part of the "Generic Icon Properties". Similarly, the "Basic Icon Properties" contains text fields for icon text, icon image location and icon type. The *iconimageURI* field specifies the icon image location inside the Java CoG Kit *resources* module. The string *NATIVE_URI* is shown inside the *iconimageURI* field if the icon is a native OS icon. The field *AppClass* specifies which class to execute in the desktop internal frame. For example one could put *javax.swing.JButton* in the *AppClass* field and specify the string, *testing*, inside the *AppClassArgs*. If after saving

these icon properties this icon was executed by double clicking, it would open a desktop internal frame containing a *JButton* with the text *testing*. Please note that the double clicking action of the NATIVE icons gives preference to the native OS executable specified in the "Generic Icon Properties" field over launching the application class specified in the "Basic Icon Properties" text field. The previous button example launched a *JButton* through its constructors. The checkbox *UseMainMethod* if checked launches the *AppClass* through its *main* method passing the string arguments specified in *AppClassArgs*. This can be used to launch external visual Java applications.

Also contained in this package is the *DesktopIconGroup* class which is a multiple icon data structure. It is used as a data flavor during drag and drop providing basic icon group methods like sorting by icon type, and retrieving all icon types from icon group.

B.4 Basis-Set Interfaces

The *Access* interfaces reside in the *interfaces* module of the *desktop* package. They form the Basis-set interfaces used by all other classes to communicate. They follow a very simple naming convention, *Access* followed by the entity being accessed. For example the *AccessIcons* interface provides method definitions needed to access icon related actions like adding and removing icons from a container like a desktop. Similarly, *AccessClose* allows a uniform interface to call when a component is being destroyed, like closing the desktop or an internal component. The following is a listing of the interfaces.

AccessActionProxy used by *CoGTop,DesktopIcon,DesktopToolBar* interfaces to participate in the Action Proxy Framework (APF) (see Subsection B.4)

AccessAttributes used by the *AttributesHolder* class in the *desktop/util* package to provide uniform access to a attribute holder data structure.

AccessClose called by the *CoGTop* interface and other Gridface modules like Directory-Browser and ShellPanel before closing.

AccessDesktop used by the *GridCommandManager* and the *DesktopToolBar* interface along with the *GCMLoggerTable* and the *CoGLogFrame* class to maintain the current desktop object reference. Since many visual components are burried within several other components, this interface provides a simple way to access the desktop.

AccessIconProperties used by the *DesktopIcon* interface to access basic Icon properties like icon text and icon id.

AccessIcons used by the *CoGTop* interface to provide uniform icon access methods like add and remove icon. These methods apply to visual containers like the desktop.

AccessImageOverlay used by the *AbstractIcon* class to enable active icon image overlays.

AccessPopup used by the *CoGTop* and the *DesktopIcon* interface and the *GCMLoggerTable* class to enable mouse popup for those elements.

AccessPreferences used by *CoGTop, DesktopIcon* and *DesktopToolBar* interfaces and *AbstractDesktopContainer, AttributesHolder* and *ObjectPair* classes to enable loading and saving desktop element attributes to and from XML file.

AccessPropertiesPanel used by the *DesktopIcon* interface to access the desktop element properties panel, like the icon properties dialog panel.

AccessSaveChanges used by the *DesktopInternalFrame* interface and the *AbstractDesktopContainer* class to enable or disable user notifications before closing visual containers like the internal frames and the desktop frame.

AccessToolBar used by the *CoGTop* interface and the *AbstractDesktopContainer* class to provide uniform access to the desktop toolbar.

Action Proxy Framework

In addition to the *Access* interfaces, the desktop interfaces also include the Action Proxy Framework (APF) interfaces. The APF assist drag-n-drop and other mouse events for the desktop. It abstracts user actions and delegates responsibility for those actions to the implementing classes, providing a flexible user interaction framework. The APF is strongly embedded in the *DesktopTransferHandler* class, which handles all drag-n-drop related activities (see Section B.1). Currently, the APF supports three actions, two related to drag-n-drop and one for handling the mouse click event. The *CanImportActionProxy* and the *ImportDataActionProxy* interfaces are used by the *DesktopTransferHandler* to first check if the APF component can accept the drag-n-drop before delegating the drop action. For example, both the desktop and the desktop icons are APF components which implement both *CanImportActionProxy* and the *ImportDataActionProxy* interfaces. When user drags an icon the *DesktopTransferHandler* checks to see if the dragged over component supports that drag by invoking the *CanImportActionProxy* interface, if it does not then the user is visually notified. If the component being dragged over does support the drop then the *ImportDataActionProxy* is called for that drop component to complete the drag-n-drop action. The drag-n-drop APF framework is currently being used by the desktop, icons, and the toolbar. Similarly, the *MouseActionProxy* enables delegation to a mouse click event. This feature is used by the icons to detect mouse clicks on top of the icons. The UML class diagram for this framework is depicted in Figure 20.

B.5 Dynamic Form Panel (DFP)

The Dynamic Form Panel (DFP) framework extends the *FormPanel* interface providing a quick and simple visual form creation methodology. The DFP dynamically invokes specified get methods on an underlying object to retrieve string values, which then are rendered inside the form panel and presented to the user. Once the user confirms changes in the form

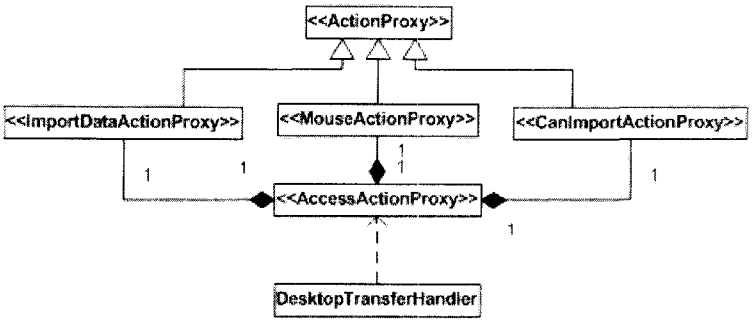

Figure 20: Action Proxy Framework

text fields, the DFP invokes set methods to transition form changes to the underlying object. Through the *FormPanelSet* class, the DFP has the ability to perform these set and get methods on several underlying objects with different set and get method calls. The desktop icons use the DFP to control icon attributes through the icon properties panel.

Suppose using the DFP we want to edit the *name* property of an object of class *SimpleObject* containing methods: *getName* and *setName*. Invoking the following code within the *SimpleObject* we would use the DFP in the following manner.

```
SimpleFormPanel sfp = new SimpleFormPanel("SimpleObject Properties");
ArrayList keys = new ArrayList();
keys.add("Name");
sfp.load(keys, this); //this specifies which object to invoke
```

Once the user confirms the changes through the DFP text fields we invoke the following to transport form changes to the underlying *SimpleObject*.

```
sfp.export();
```

Figure 21 depicts the DFP framework.

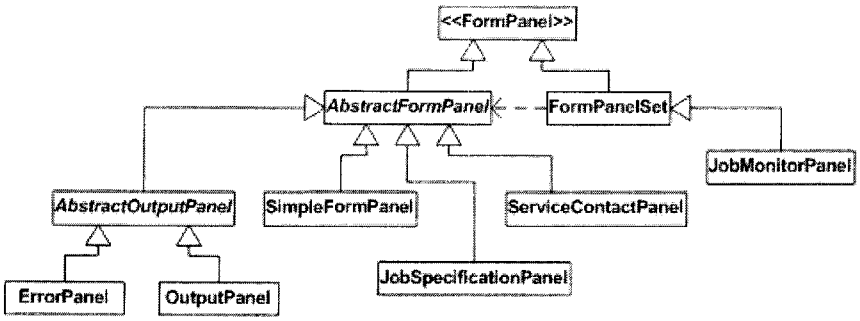

Figure 21: Dynamic Form Panel (DFP) Framework

B.6 Desktop Toolbar

The *DesktopToolBarImpl* class implements the *DesktopToolBar* interface. The toolbar serves as a docking station for icons, in addition to being the default SYSTEM icon container. The toolbar state can be saved through the desktop preferences as defined in Section B.8.

B.7 Desktop Utilities

Several utility classes are provided by the *util* sub-package.

AttributesHolder class is used by the desktop icons to access dynamic icon attributes.

ObjectPair is used to store a pair of objects. The Grid Desktop uses this to store the application class name and the related object used to invoke desktop internal frames.

DesktopProperties extends the *java.util.Properties* class to provide a central management of desktop properties including desktop preferences as explained in Section B.8.

DesktopUtilities is a static class providing implementation for simple desktop tasks like converting a *List* object to a *String*.

B.8 Desktop Preferences

The Grid Desktop can save and load icon properties to an XML file using the *java.util.Preferences* package, enabling desktop users to port workspace from one system to another (see Figure 22). The desktop icons being saved can be situated either on the desktop or the toolbar, as the responsibility of saving preferences is delegated through the *AccessPreferences* interface. The Desktop Preferences schema format is outlined in Figure 23, and a sample desktop XML file is provided in the Appendix.

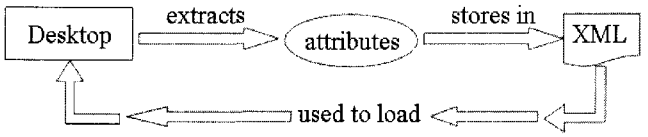

Figure 22: Desktop Preferences Simplified

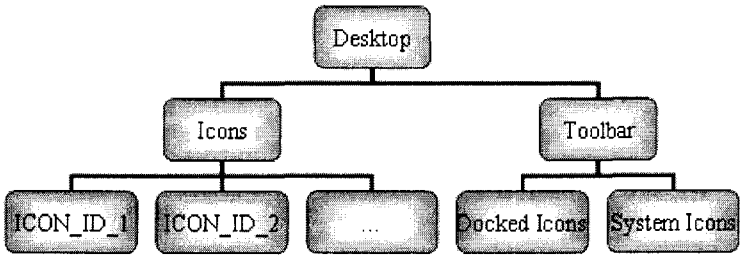

Figure 23: Desktop Preferences Schema

C. Implementation

The Grid Desktop, also referred to as *CoGTop*, is a user centric grid workspace that extends the abstract desktop framework. Its origins can be traced to the time when even the term Grid was not yet invented [102, 103, 104]. The Grid Desktop bundles modules already available in the Java CoG Kit into a single easy to use, persistent, and portable workspace. In addition, the Grid Desktop provides several Grid centric icon types like

JOB_SUBMISSION, JOB_SPECIFICATION and SERVICE. Also, native OS RSL files specifying RSL specifications [105] can be placed on to the Grid Desktop, and later dropped on a JOB_SUBMISSION or a JOB_SPECIFICATION icon type to invoke a job submission on a service. All icon properties are edited through their properties form.

Figure 24 depicts a screen shot of the desktop anotated with labels of a subset of desktop components.

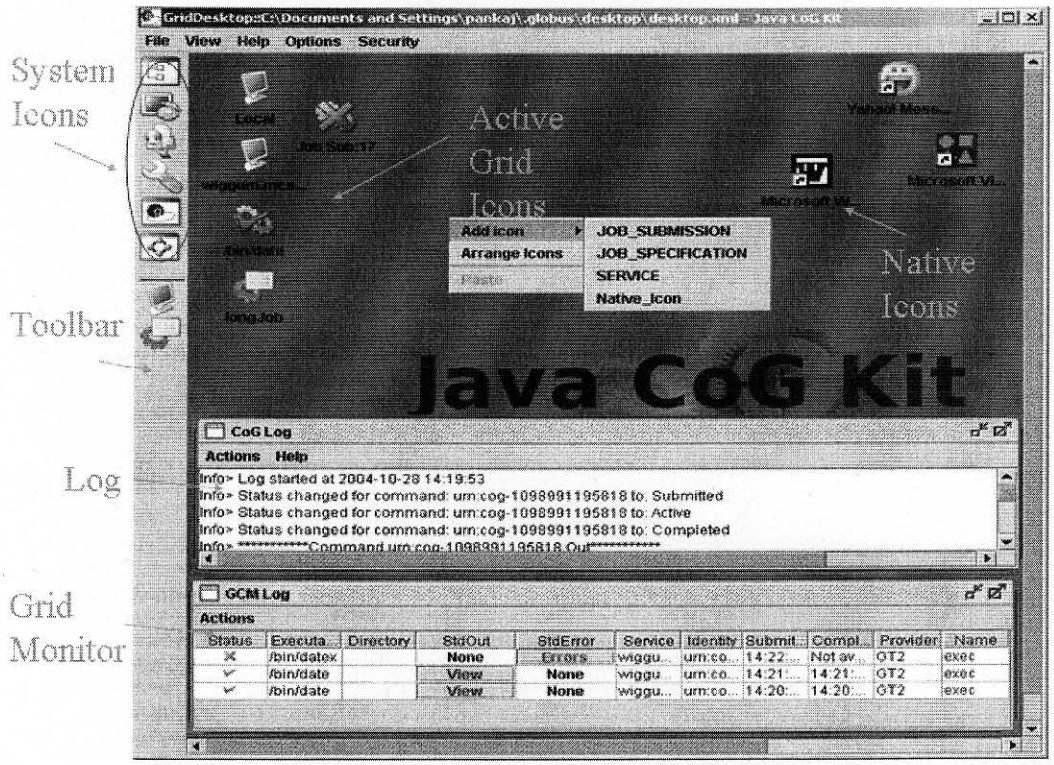

Figure 24: Desktop Overview

C.1 Underlying Components

The Java CoG Kit Grid Desktop relies on several underlying components and modules. The following is a list of components in order of dependency, starting with the Grid Desktop to the underlying Grid environment.

- Grid Desktop
- GridFaces
- Grid Command Manager (GCM)
- Java CoG Kit Karajan Workflow
- Java CoG Kit Abstractions
- Grid environment (Globus, Condor, etc.)

As the Grid Desktop is part of the *Gridfaces* module, it incorporates the Grid Shell and the Directory Browser component as part of the desktop. The Grid Shell component provides a shell environment, similar to UNIX shells, that functions over Grid systems. Users of the Grid Shell can perform many basic and advanced Grid functions like file operations and file transfers. The Directory Browser is a similar component with an enhanced visual interface. It looks and functions a lot like the Microsoft Windows Explorer, supporting drag-n-drop of files and directories between Grid systems. More information can be found about these and other Gridface components in the Java CoG Kit 4 manual [4].

The core of every Gridface element, including the desktop, is a single Grid Command Manager (GCM) that controls all Grid related activity between the Gridface components and the rest of the Java CoG Kit. Every job submission, file transfer, and file operation task is initiated through the *AccessGCM* interface and tracked by the GCM throughout its life cycle. The GCM monitor table is provided to visualize these Grid activities, as explained in Section C.7.

The Java CoG Kit Karajan visual workflow editor can be invoked as an internal desktop frame, adding a fully functional and highly sophisticated workflow system into the desktop that can handle larger coordinated tasks. The core Java CoG Kit module is the abstractions module, which provides a seamless interface for Grid activities that stays homogeneous

despite of the underlying Grid systems. This hides Grid system complexities from the application user, who can then spend time on more important activities. The abstractions module is the core module for all Grid Desktop activities requiring Grid actions. More information about the Karajan system and the Java CoG Kit abstractions can be accessed through the Java CoG Kit manual [4].

C.2 Prerequisites

Before starting the Grid Desktop the user must make sure the network connection is active and can connect to the Grid. In addition, follow the procedures as explained in [4] to set up the Java CoG Kit. In case the Grid is behind a firewall, use VPN to connect to the internal network by following these steps. Note that if you reconfigure the network settings from within the desktop, you must restart the Grid Desktop for the changes to take effect.

1. Log in to VPN

2. After establishing the connection run the Java CoG Kit setup component and re-probe the CoG properties IP address to match the VPN connection IP address.

```
> cog-setup
```

Once the connection is validated to the Grid, make sure the *.globus* directory is there in user home directory path. Failing to have this directory in its proper path will keep the Grid Desktop from starting. Inside the *.globus* directory the desktop configuration file exists, called *desktop.properties*.

```
$HOME/.globus/desktop.properties
```

In this file, one can specify several desktop attributes used to start the Grid Desktop.

desktopfile specifies where the default desktop preferences file is located.

maxwidth specifies maximum width for the desktop, fitting all icons and internal frames accordingly.

maxheight specifies maximum height for the desktop, fitting all icons and internal frames accordingly.

If a *desktop.properties* file does not exist, then a default file will be created using the *DesktopProperties* utility class. In it the *desktopfile* attribute will point towards *.globus/desktop/desktop.xml*, which will be created if not present. Also the *maxwidth* and *maxheight* will be set to the maximum dimensions of the current screen size.

An example *desktop.properties* file is given below.

```
#Java CoG Kit Desktop Configuration File
#2004-11-01 14:52:28
desktopfile=C:\Documents and Settings\pankaj\.globus\desktop\desktop.xml
maxwidth=1024
maxheight=768
```

C.3 Launching

The Grid Desktop is launched by executing

```
> cog-desktop
```

The usage instructions for starting the desktop are provided by the Java CoG Kit manual page available in Appendix IV.

C.4 Grid Command Manager Preferences

Adding to the XML export function from the abstract desktop framework, the Grid Desktop also partially supports exporting GCM tasks and task graphs to a XML file. The idea is to save on going Grid tasks to be started at a later time. This functionality was developed by another team member, and is not thoroughly tested through the desktop. It is recommended to wait for future Java CoG Kit releases to use this functionality.

C.5 Icons

Building on the abstract desktop framework, the Grid Desktop adds three more icon types. Both the abstract desktop framework and Grid Desktop icon types are summarized below.

JOB_SPECIFICATION is based on the RSL specification, where the executable, working remote directory, standard input and standard output can be specified. This icon type should be dropped on a SERVICE icon to execute the specification on that service.

JOB_SUBMISSION contains a complete description of a job, including service contact and job specification information. This icon type is capable of executing jobs through its popup and its properties frame. Similar to the JOB_SPECIFICATION icon type, this icon type can also be dropped on a SERVICE icon to execute tasks. The job output can be viewed through the Grid Monitor Frame or by double clicking the icon, once task is completed.

SERVICE contains service contact information and accepts drag and drop from both JOB_SPECIFICATION and JOB_SUBMISSION icons, initiating job submissions specified in the service contact.

NATIVE represents a native file, which can be executed by double clicking. This icon is also created by dropping a native Operating System file onto the desktop.

SYSTEM Represent preconfigured icons like the Directory Browser and Grid Shell. These icons cannot be added or edited from both the desktop or the preferences XML file.

These icons can be added to the desktop interactively, just like other popular desktop environments. Right click on the desktop and navigate to "Add icon" menu and select the icon type to add to the desktop.

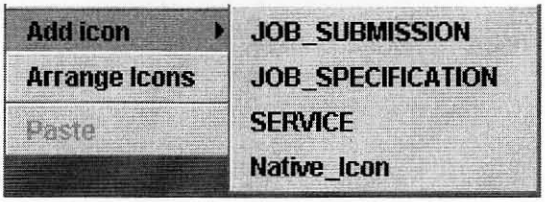

Figure 25: Add Icon Menu

Once added, the individual icon properties can be configured by following to the "Properties" menu item contained in each icon popup. For convenience sake, the SERVICE icon and JOB_SPECIFICATION icons can access the properties frame by double-clicking the icon also.

All Grid icons support drag-n-drop, and also provide flexibility to accept or reject drops depending on specific icon type pairs. For instance a SERVICE icon cannot be dropped on to NATIVE icons, as intuitively it does not make sense. On the other hand, a JOB_SUBMISSION icon can be dropped on to a SERVICE but not on a NATIVE icon. This relation is specified through a Boolean matrix, which is part of the *GridIconImpl* class.

```
boolean[][] canImportMatrix = {
        //[SERVICE,JOB_SUBMISSION,JOB_SPECIFICATION] (Drop Icon)
        {false,false,false}, //SERVICE (Drag Icon)
        {true,true,false},  //JOB_SUBMISSION (Drag Icon)
        {true,false,true},  //JOB_SPECIFICATION (Drag Icon)
        {false,true,true}  //NATIVE – RSL FILE(Drag Icon)
};
```

JOB_SUBMISSION and JOB_SPECIFICATION are active icons, meaning their job status is reflected visually through their icon image. A visual tour of the icon image overlays and their meanings follow.

Submitting a job through the icons can be done in two ways. First, by dragging and dropping a JOB_SUBMISSION or a JOB_SPECIFICATION icon onto a SERVICE icon. Second, by directly executing the JOB_SUBMISSION icon through the icon popup "Run Task" method or by using the "Execute" button in the JOB_SUBMISSION proper-

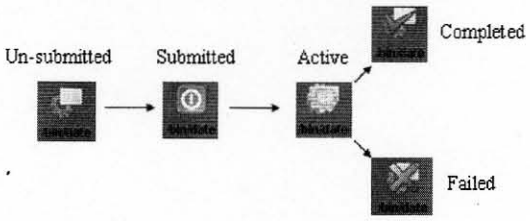

Figure 26: Action Icon States

ties frame. A SERVICE icon is not necessary for JOB_SUBMISSION icons as they also contain service contact information.

C.6 CoGTop Log Frame

The Desktop Log frame logs all desktop activity to a scrollable text area. The logs being displayed can be controlled by setting a higher or lower Log Level. This can be done through the Desktop Log frame menu bar. The log can also be exported to a file using the "Save Log" action from the Log frame menu bar. This will save all log activity regardless of the current Log Level. Additionally, the "Clear" log action clears all log information from the frame.

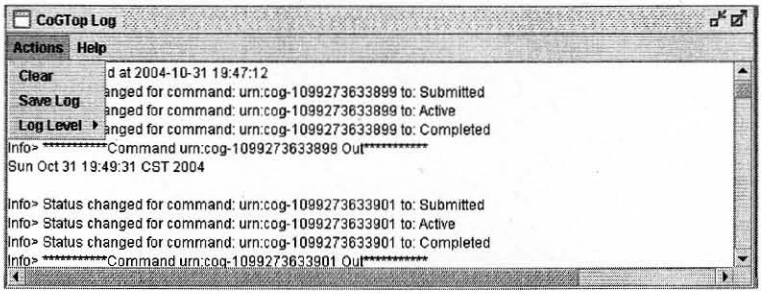

Figure 27: CoGTop Log Frame

C.7 Grid Monitor Frame

The Grid Monitor frame displays Grid task data in a table that includes status, job executable name, directory location, job submit time and job completion time. It also includes

additional features such as viewing job output and job errors by clicking the respectable buttons displayed in the monitor frame, or by hovering the mouse over the respected buttons in the table.

Each row in the table represents a Grid Command, and currently supports canceling individual commands by invoking cancel action through row right click popup. Other actions such as Grid Command suspend, resume and remove will be added in future releases as the GCM does not fully support them yet.

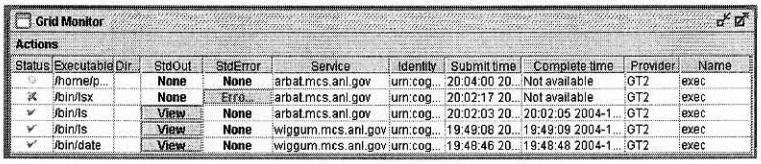

Figure 28: Grid Monitor Frame

D. Evaluation and Future Extensions

Thus far we have seen how the Grid Desktop functions both as a desktop paradigm and as a Grid portal. In this section we shall evaluate the Grid Desktop by noting its limitations, recognizing its maintainance issues and providing possible future extensions.

D.1 Limitations

The Grid Desktop tries to simulate popular desktop paradigms like Windows and KDE. Both of which have been in developement for several years, involving several man hours and careful planning. Although they provide a good implementation for the desktop paradigm, they do not provide any support for Grid enabled environments. Grid Desktop attempts to balance both desktop paradigm and Grid requirements into a single, portable, and homogeneous system. While achieving these goals the Grid Desktop contains a few limitations.

Usability The Grid Desktop does not offer all attributes of the desktop paradigm like startup services, network services and a wide range of internal applications. Though

it does provide active icons and drag-n-drop patterns.

Memory The Grid Desktop runs inside the Java Virtual Memory (JVM) and depends on memory specifications of the local system. If the Grid Desktop submits hundreds of jobs and monitors them through the Grid monitor table, the JVM might reach its capacity, causing a performance deficiency. These limitations are reserved for future research which may test the performance quantitatively, and address issues regarding scalability.

Grid functionality The Grid Desktop is ultimately limited to offer Grid functionality provided by the Java CoG Kit abstractions. If the Java CoG Kit abstractions were to break, the Grid Desktop would also follow.

Grid Command Manager (GCM) When the Grid Desktop interacts with Grid Environments it goes through the GCM. It should be noted that the GCM component was not fully developed at the time the Grid Desktop was implemented. Thus, potentially several Grid functionality issues like task suspend, resume, and checkpoint, are limited.

D.2 Maintainance

As part of the Java CoG Kit Bugzilla system [106], the Grid Desktop maintains a bug listing that can be viewed and updated by community members. This enables a wide range of testing, support, feedback, and documentation opportunities. The Bugzilla system is also accessible from one of the Grid Desktop SYSTEM icons on the toolbar. This is provided to give community users instant access to bug reporting.

D.3 Future Extensions

Possible Grid Desktop extensions can be classified into two categories: enhancements and Grid functionality additions. In this section we shall examine only the extensions con-

tributed by the author, while other extensions can be viewed through the Java CoG Kit Bugzilla system [106]. The enhancements could be summarized as the following.

- Dynamically resize desktop size once started. This will help during projector presentations, which could require a smaller resolution.
- Integrate the Access Grid software into the Grid Desktop.
- Enhance Directory Browser user experience by caching contents.
- For the CoG Log window, add font colors for different log levels.
- Send email notifications once a Grid icon has changed its status.
- Be able to visually select an icon image through icon properties panel.
- Be able to select and group multiple icons on the desktop.
- Provide an active desktop wallpaper similar to the IBM Ambient active desktop [107]
- Port desktop gridface to handheld devices like PDA and Cell phones [108].
- Enhance the view for the GCM Log table by providing views like Table-Tree which can also show collapsable task graphs.

Grid functionalities can be summarized as the following.

- Ability to save and load menu bar, menu items and internal frames as part of the Desktop Preferences.
- The GCM needs to be developed to support save and load task status data as part of the Desktop Preferences.
- The *desktop.properties* file should be modifiable from the Java CoG Kit setup, to centralize the Java CoG Kit configurations.

- Support for multiple desktops.

All of the above enhancements and functionalities are good additions, but the most significant Grid Desktop addition would be the ability to close the desktop completely without suspending or canceling any active jobs. Currently, the Grid Desktop works on a synchronous Grid environment that requires fault free connection to the Grid, but if the connection is lost, the Grid Desktop breaks and looses all active and suspended jobs. Once this functionality is added to the Java CoG Kit abstractions, the Grid Desktop can leverage a fully functional, portable, and fault tolerant Grid interface.

In addition to the above suggesstions we could also explore tailoring Grid Desktop to provide built in tools for several Grid related problems like Data Mining, Bioinformatics, Finance, Homeland security, Astronomy, etc. This would bring an appeal for the Grid Desktop to the novice Grid users originating from a wide range of scientific studies.

IV. CONCLUSION

In conclusion, the Java CoG Kit Grid Desktop has contributed in three ways to the Grid Community. First by developing a simple and easy to use Grid interface replicating the widely popular desktop paradigms like Windows and KDE. Unlike several Grid desktop paradigms which emerged later, the Java CoG Kit Grid Desktop provides a unique, flexible, and functional interface for a wide range of underlying Grid technologies, addressing several Human Computer Interaction (HCI) and Problem Solving Environment (PSE) issues. In turn, serving both the novice and advanced Grid users by enabling single drag-n-drop Grid task submissions and also advanced workflow task submissions through the Karajan engine. Along with the implementation, the Grid Desktop also provides documentation as part of the Java CoG Kit manual [4], which serves as a guide for using the desktop and noting future extensions. The infrastructure for continuing development is very important, especially when part of the widely popular Java CoG Kit open source project. Finally, the Grid Desktop addresses maintainance issues through the Java CoG Kit Bugzilla system. Here through community feedback, the developers and users sustain dialogues on both problems and enhancements for the Grid Desktop.

REFERENCES

[1] G. von Laszewski and K. Amin, *Grid Middleware*. Wiley, 2004, ch. Chapter 5 in Middleware for Commnications, pp. 109–130. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--grid-middleware.pdf

[2] G. von Laszewski and P. Wagstrom, *Tools and Environments for Parallel and Distributed Computing*, ser. Series on Parallel and Distributed Computing. Wiley, 2004, ch. Gestalt of the Grid, pp. 149–187. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--gestalt.pdf

[3] K. Amin, G. von Laszewski, R. A. Ali, O. Rana, and D. Walker, "An Abstraction Model for a Grid Execution Framework," *Euromicro Journal of Systems Architecture*, 2004, accepted for publication.

[4] G. von Laszewski, "The Java CoG Kit 4.0a 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, Oct. 14 2004. [Online]. Available: http://www.cogkit.org/doc/cog-manual-4-0-a.pdf

[5] K. Amin, M. Hategan, G. von Laszewski, N. J. Zaluzec, S. Hampton, and A. Rossi, "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

[6] "Mike Hategan and Gregor von Laszewski, personal communication."

[7] "Commodity Grid Kits," Web Page. [Online]. Available: http://www.cogkits.org

[8] G. von Laszewski, M. W. Bone, I. Fatima, M. Sosonkin, R. Winch, N. N. Vijayakumar, P. Sahasrabudhe, K. Amin, M. Hategan1, J. DiCarlo, and D. Angulo, "Towards the development of a bioinformatics grid desktop," Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, U.S.A., Preprint ANL/MCS-ANL/MCS-P1189-0804, Aug. 2004, in partial fulfillment of the REU 2004 Site on Grid Computing and Bioinformatics. [Online]. Available: http://www.cogkit.org

[9] "Argonne National Laboratory - Pioneering Science and Technology," Web Page. [Online]. Available: http://www.anl.gov

[10] A. Abbas and A. Abbas, *Grid Computing: A Practical Guide to Technology and Applications*. Charles River Media, Inc., 2003.

[11] 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

[12] I. Foster, "The Grid: A New Infrastructure for 21st Century Science," *Physics Today*, vol. 55, no. 22, p. 42, 2002. [Online]. Available: http://www.aip.org/pt/vol-55/iss-2/p42.html

[13] H. Casanova, "Distributed computing research issues in grid computing," *SIGACT News*, vol. 33, no. 3, pp. 50–70, 2002.

[14] "Ebay Online Auction," Web Page. [Online]. Available: http://www.ebay.com

[15] E. Gallopoulos, E. Houstis, and J. R. Rice, "Computer as Thinker/Doer: Problem-Solving Environments for Computational Science," *IEEE Comput. Sci. Eng.*, vol. 1, no. 2, pp. 11–23, 1994.

[16] J. R. Rice and R. F. Boisvert, "From Scientific Software Libraries to Problem-Solving Environments," *IEEE Comput. Sci. Eng.*, vol. 3, no. 3, pp. 44–53, 1996.

[17] D. Abramson, J. Dongarra, E. Meek, P. Roe, and Z. Shi, "Simplified Grid Computing through Spreadsheets and NetSolve," in *Proceedings of the High Performance Computing and Grid in Asia Pacific Region, Seventh International Conference on (HPCAsia'04)*. IEEE Computer Society, 2004, pp. 19–24.

[18] 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.

[19] "Netsolve Web Page," 2001. [Online]. Available: http://www.cs.utk.edu/netsolve

[20] Y. Kim, I. Ra, B. Kim, S. Hariri, and H. W. Park, "A Grid-enabled Adaptive Problem-Solving Environment ," in *2nd European Across Grids Conference (AxGrids 2004) Proceedings*, Nicosia, Cyprus, 28 Jan.-30 Jan. 2004. [Online]. Available: http://grid.ucy.ac.cy/axgrids04/AxGrids/papers/E00-555793154.pdf

[21] D. Becker and P. Merkey, "The Beowulf Project," http://www.beowulf.org/, 2002. [Online]. Available: http://www.beowulf.org/

[22] A. Barak, "The Mosix Homepage," Web Page, 2002. [Online]. Available: http://www.mosix.org/

[23] 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. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--cog-pse-final.pdf

[24] "HCI Bibliography : Human-Computer Interaction / User Interface Usability," Web Page. [Online]. Available: http://www.hcibib.org/

[25] J. Preece, Y. Rogers, H. Sharp, and D. Benyon, *Human-Computer Interaction*. Addison-Wesley Longman Ltd., 1994.

[26] A. K. Noor, "Computing technology: frontiers and beyond," pp. 1–23, 2002.

[27] J. George Chin, L. R. Leung, K. Schuchardt, and D. Gracio, "New paradigms in problem solving environments for scientific computing," in *Proceedings of the 7th international conference on Intelligent user interfaces*. ACM Press, 2002, pp. 39–46.

[28] M. Corporation, *The Micorsoft Windows User Experience*. Redmond, WA: Microsoft Press, 1999.

[29] "Sun Java Desktop System," Web Page. [Online]. Available: http://wwws.sun.com/software/javadesktopsystem

[30] "Open Group - Desktop Technologies," Web Page. [Online]. Available: http://www.opengroup.org/desktop/

[31] C. D. Group, *Common Desktop Environment 1.0*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[32] A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human-Computer Interaction*, 2nd ed. Hillsdale, NJ: Prentice Hall, 1998.

[33] J. M. Schopf and B. Nitzberg, "Grids: The Top Ten Questions," *Scientific Programming*, vol. 10, no. 2, pp. 103–111, August 2002. [Online]. Available: http://www-unix.mcs.anl.gov/~schopf/Pubs/topten.final.pdf

[34] I. Foster, "What is the Grid? A Three Point Checklist ," 22 July 2002. [Online]. Available: http://www.gridtoday.com/02/0722/100136.html

[35] "GRIDtoday: Daily News and Information For The Global Grid Community," Web Page. [Online]. Available: http://www.gridtoday.com/gridtoday.html

[36] "Grid Computing Info Centre (GRID Infoware)," Web Page. [Online]. Available: http://www.gridcomputing.com/

[37] O. F. Rana, A. Shaikhali, and G. von Laszewski, *Grid Computing: A Practical Guide to Technology and Applications*. Hingham, MA: Charles River Media, 2003, ch. Creating and Managing Grid Services, pp. 189–223.

[38] M. Calleja, L. Blanshard, C. C. Richard Bruin, A. Thandavan, R. Tyer, P. Wilson, V. Alexandrov, R. J. Allen, J. Brodholt, M. T. Dove, W. Emmerich, and K. K. van Dam, "Grid tool integration within the eMinerals project," in *Proceedings of the UK e-Science All Hands Meeting 2004*, Nottingham, UK, 31 Aug.-3 Sept. 2004, pp. 812–817. [Online]. Available: http://www.allhands.org.uk/proceedings/proceedings/proceedings.pdf

[39] 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

[40] "The Globus Project," Web Page. [Online]. Available: http://www.globus.org

[41] I. Foster, "Internet Computing and the Emerging Grid," *Nature*, 7 Dec. 2000. [Online]. Available: http://esc.dl.ac.uk/StarterKit/nature.html

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

[43] "GRAM Job Manager Reference Manual."

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

[45] "Globus: Acquiring User and Host GSI Certificates," Web Page. [Online]. Available: http://www.globus.org/security/v1.1/certs.html

[46] "The GridFTP Protocol and Software," Web Page. [Online]. Available: http://www.globus.org/datagrid/gridftp.html

[47] N. Karonis, "MPICH-G2 Web Page," Web Page, 2001. [Online]. Available: http://www.hpclab.niu.edu/mpi/

[48] "IBM Grid Computing," Web Page. [Online]. Available: http://www-1.ibm.com/grid/

[49] B. S. White, M. Walker, M. Humphrey, and A. Grimshaw, "LegionFS: A Secure and Scalable File System Support Cross-Domain High Performance Applications," in *Proceedings of Supercomputing 2001*, Denver, Colorado, USA, November 2001. [Online]. Available: http://legion.virginia.edu/papers/SC2001.pdf

[50] A. S. Grimshaw, M. A. Humphrey, and A. Natrajan, "A philosophical and technical comparison of Legion and Globus," *IBM J. Res. Dev.*, vol. 48, no. 2, pp. 233–254, 2004.

[51] "Condor: High Throughput Computing," Web Page. [Online]. Available: http://www.cs.wisc.edu/condor/

[52] 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.

[53] R. D. Stevens, A. J. Robinson, and C. A. Goble, " myGrid: personalised bioinformatics on the information grid," *Bioinformatics*, vol. 19, pp. 302–304, Feb. 2003. [Online]. Available: http://www.mygrid.org.uk/

[54] "The National Center for Biotechnology Information (NCBI) BLAST," Web Page. [Online]. Available: http://www.ncbi.nlm.nih.gov/BLAST/

[55] "Washington University BLAST," Web Page. [Online]. Available: http://blast.wustl.edu

[56] "European Molecular Biology Open Source Software Suite (EMBOSS) Homepage," Web Page. [Online]. Available: http://www.hgmp.mrc.ac.uk/Software/EMBOSS/

[57] "Universal Description, Discovery and Integration of Business for the Web," Web Page. [Online]. Available: http://www.uddi.org/

[58] R. Stevens, H. Tipney, C. Wroe, T. Oinn, M. Senger, P. Lord, C. Goble, A. Brass, and M. Tassabehjib, "Genome Science performed with e-Science Tools," in *Proceedings of the UK e-Science All Hands Meeting 2004*, Nottingham, UK, 31 Aug.-3 Sept. 2004, pp. 768–775. [Online]. Available: http://www.allhands.org.uk/proceedings/proceedings/proceedings.pdf

[59] R. V. van Nieuwpoort, J. Maassen, R. Hofman, T. Kielmann, and H. E. Bal, "Ibis: an efficient Java-based grid programming environment," in *Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande*. ACM Press, 2002, pp. 18–27.

[60] "The Commodity Grid Project," Web Page. [Online]. Available: http://www.globus.org/cog

[61] G. von Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke, "CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids," in *ACM Java Grande 2000 Conference*, San Francisco, CA, 3-5 June 2000, pp. 97–106. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--cog-final.pdf

[62] "Super Computing 2004 Conference," November 2004. [Online]. Available: http://www.sc-conference.org/sc2004

[63] W. MD and L. M., "BioMOBY: an open source biological web services proposal," *Brief Bioinform*, vol. 3, pp. 331–341, Dec. 2002.

[64] R. Grim, , and M. ter Linden, "GridAssist Technical Whitepaper," Dutch Space, Tech. Rep., nov 2004. [Online]. Available: http://tphon.dutchspace.nl/grease/public/GridAssist_whitepaper_technical_final_1.0-2004_October_(web).pdf

[65] J. Brown, C. Ferner, T. Hudson, A. Stapleton, R. Vetter, A. Martin, J. Martin, A. Rawls, B. Shipman, and M. Wood, "GridNexus: A Grid Services Scientific Workflow System," Nov. 2004, to be published. [Online]. Available: http://torvalds.bearlabs.uncw.edu/~awr8543/gridnexus/_files/IPDPS2005GridNexus.pdf

[66] "The Taverna Project," Web Page. [Online]. Available: http://taverna.sourceforge.net

[67] "Freeflou Project," Web Page. [Online]. Available: http://freefluo.sourceforge.net

[68] M. Chagoyen, M. Kurul, P. De-Alarcn, J. Carazo, and A. Gupta, "Designing and executing scientific workflows with a programmable integrator," *Bioinformatics*, 2004, to be published.

[69] P. M, Y. I, and A. RB., "Modelling biological processes using workflow and Petri Net models." *Bioinformatics*, vol. 18, pp. 825–837, June 2002.

[70] "BioGrid Japan," Web Page. [Online]. Available: http://www.biogrid.jp/e/research_work/gro1/guide/index.html

[71] N. NF, C. M, F. RW, K. H, T. SW, V. J, and M. MA., "Protege-2000: An Open-source Ontology-development and Knowledge-acquisition Environment." *Proc AMIA Symp*, p. 953, 2003, processing on PubMed. [Online]. Available: http://protege.stanford.edu/

[72] "Genome Annotation Pipeline (iGAP," Web Page, sandiego SuperComputing Center. [Online]. Available: http://eol.sdsc.edu:8080/eol/index.jsp

[73] de Knikker R, G. Y, L. JL, K. AK, Y. KY, C. DW, and C. KH., "A web services choreography scenario for interoperating bioinformatics applications." *BMC Bioinformatics*, vol. 5, p. 25, Mar. 2004.

[74] A. Slominski and G. von Laszewski, "Scientific Workflows Survey," Web Page, 2004. [Online]. Available: http://www.extreme.indiana.edu/swf-survey/

[75] G. von Laszewski, J. Gawor, S. Krishnan, and K. Jackson, *Grid Computing: Making the Global Infrastructure a Reality*, ser. Communications Networking and Distributed Systems. Wiley, 2003, ch. Commodity Grid Kits - Middleware

for Building Grid Computing Environments, pp. 639–656. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--grid2002book.pdf

[76] M. Thompson, J. G. Schmidt, and P. M. Dew, "White Rose Grid Portals: Practice and Experience," in *Proceedings of the UK e-Science All Hands Meeting 2004*, Nottingham, UK, 31 Aug.-3 Sept. 2004, pp. 952–955. [Online]. Available: http://www.allhands.org.uk/proceedings/proceedings/proceedings.pdf

[77] "The Jetspeed Webpage," Web page. [Online]. Available: http://jakarta.apache.org/jetspeed/

[78] "Portal Construction Toolkit for the Grid (PCT4G)," Web Page, cCGrid2003, IEEE/ACM International Symposium on cluster computing and the Grid. [Online]. Available: http://ccgrid2003.apgrid.org/online_posters/posters/024.pdf

[79] A. Natrajan, A. Nguyen-Tuong, M. Humphrey, and A. Grimshaw, "The Legion Grid Portal," in *Grid Computing – Grid 2001*, ser. Lecture Notes in Computer Science, vol. 2242, ACM, IEEE. Denver, Colorado: Springer-Verlag, Heidelberg, Germany, November 2001. [Online]. Available: http://legion.virginia.edu/papers/HPCS01.pdf

[80] T. Suzumura, S. Matsuoka, and H. Nakada, "A Jini-based computing portal system," in *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*. ACM Press, 2001, pp. 24–24.

[81] "GridSpeed: Grid Application Portal Generator," Web Page, tokyo Institute of Technology. [Online]. Available: http://spam.sdsc.edu:8080/gridspeed/index.jsp/

[82] "BioPortal," Web Page, academia Sinica Computing Centre, Taipei, Taiwan. [Online]. Available: http://big.pcf.sinica.edu.tw/

[83] "Grid Portal Interface for Interactive Use and Monitoring of High-Throughput Proteome Annotation," Web Page. [Online]. Available: http://matsu-www.is.titech.ac.jp/paper/suzumura/LSGW_EOL_GRIDPORTAL.final.pdf/

[84] "Bioinformatics Portal using OGSA," Web Page. [Online]. Available: http://ntu-cg.ntu.edu.sg/Grid_competition/report/grid-6.pdf

[85] "Biobar," Web Page. [Online]. Available: http://biobar.mozdev.org

[86] C. T. et al., "The Helmholtz Network for Bioinformatics: an integrative web portal for bioinformatics resources." *Bioinformatics*, vol. 20, pp. 268–270, Jan. 2004.

[87] G. SH, L. YT, B. GG, and M. JL., "Hembase: browser and genome portal for hematology and erythroid biology." *Nucleic Acids Res*, vol. 32, pp. 572–574, Jan. 2004.

[88] "Open Grid Computing Environments," Web Page. [Online]. Available: http://www.ogce.org

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

[90] "NPACI HotPage," Web Page, 2001. [Online]. Available: https://hotpage.npaci.edu/

[91] O. TM., "Talisman–rapid application development for the grid." *Bioinformatics*, vol. 19, pp. 212–214, 2003. [Online]. Available: http://talisman.sourceforge.net/

[92] "Entrada," Web Page. [Online]. Available: http://www-unix.mcs.anl.gov/~gose/entrada/

[93] "The CogBox," Web Page. [Online]. Available: http://www.extreme.indiana.edu/~btemko/cogbox/index.html

[94] A. Reinefeld, F. Schintke, and G. Din, "GuiGen: a toolset for creating customized interfaces for grid user communities," *Future Gener. Comput. Syst.*, vol. 18, no. 8, pp. 1075–1084, 2002.

[95] "SETI@Home," Web Page, Feb. 2002. [Online]. Available: http://setiathome.ssl.berekeley.edu/

[96] "Folding@Home Distributed Computing," Web Page. [Online]. Available: http://www.stanford.edu/group/pandegroup/folding/

[97] "FightAIDS@Home," Web Page. [Online]. Available: http://fightaidsathome.scripps.edu/

[98] "KDE desktop," Web Page. [Online]. Available: http://www.kde.org

[99] "ZerahStar Zesktop," Web Page. [Online]. Available: http://www.zerahstar.com/

[100] M. Foster, D. Hanlon, J. MacLaren, J. Marsh, S. Pettifer, and S. Pickles, "Grid-Enabled Desktop Environments: The GRENADE Project," in *Proceedings of the UK e-Science All Hands Meeting 2004*, Nottingham, UK, 31 Aug.-3 Sept. 2004, pp. 912–919. [Online]. Available: http://www.allhands.org.uk/proceedings/proceedings/proceedings.pdf

[101] "Migrating Desktop and Roaming Access," Web page, Poznan Supercomputing and Networking Center,Poznan, Poland, April 2004. [Online]. Available: http://www.man.poznan.pl/coe/documents/Migrating_Desktop_WP.pdf

[102] G. von Laszewski, M. Seablom, M. Makivic, P. Lyster, and S. Ranka, "Design Issues for the Parallelization of an Optimal Interpolation Algorithm," in *Coming of Age, Proceedings of the 4th Workshop on the Use of Parallel Processing in Atmospheric Science*, G.-R. Hoffman and N. Kreitz, Eds., European Centre for Medium Weather Forecast. Reading, UK: World Scientific, 21-25 Nov. 1994, pp. 290–302. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski94-4dda-design.pdf

[103] 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, (The initial version of this paper was available in 1996). [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--CooperatingJobs.ps

[104] 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

[105] "Resource Specification Language," Web Page, 2002, http://www.globus.org/gram/gram_rsl_parameters.html.

[106] T. G. Project, "The Java CoG Kit Bug List," Web Page, 2004. [Online]. Available: http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=Java+CoG+Kit

[107] "Ambient Sametime Active Desktop," Web Page. [Online]. Available: http://www-128.ibm.com/developerworks/lotus/library/article/ambientST/

[108] S. P. Nee and R. S. Kalawsky, "Developing a Roaming PDA-Based Interface for a Steering Client for OGSI::Lite using .Net: Practical Lessons Learned," in *Proceedings of the UK e-Science All Hands Meeting 2004*, Nottingham, UK, 31 Aug.-3 Sept. 2004, pp. 587–592. [Online]. Available: http://www.allhands.org.uk/proceedings/proceedings/proceedings.pdf

APPENDIX A

Sample Desktop Preferences XML File

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE preferences SYSTEM 'http://java.sun.com/dtd/preferences.dtd'>

<preferences EXTERNAL_XML_VERSION="1.0">
  <root type="user">
    <map />
    <node name="Desktop_1__1099111886827">
      <map>
        <entry key="frame.title" value="New Desktop" />
        <entry key="frame.class"
         value="org.globus.cog.gridface.impl.desktop.GridDesktopContainer" />
        <entry key="frame.width" value="924" />
        <entry key="frame.height" value="661" />
        <entry key="frame.x" value="50" />
        <entry key="frame.y" value="57" />
        <entry key="icon.count" value="1" />
      </map>
      <node name="icons">
        <map />
        <node name="ICON_ID_15">
          <map>
            <entry key="icon.name" value="/bin/date" />
            <entry key="icon.x" value="8" />
            <entry key="icon.y" value="6" />
            <entry key="icon.type"
             value="org.globus.cog.gridface.impl.desktop
             .icons.GridIconImpl:JOB_SPECIFICATION" />
            <entry key="icon.imageURI" value="images/32x32/co/jobspec-icon.png" />
            <entry key="icon.launchStaticMain" value="false" />
            <entry key="#ATTRIB:directory" value="/tmp" />
            <entry key="#ATTRIB:stderror" value="errorFile" />
            <entry key="#ATTRIB:executable" value="/bin/date" />
            <entry key="#ATTRIB:batchjob" value="false" />
            <entry key="#ATTRIB:localexecutable" value="false" />
            <entry key="#ATTRIB:redirected" value="true" />
            <entry key="#ATTRIB:stdoutput" value="outputFile" />
          </map>
        </node>
      </node>
```

```
<node name="toolbar">
  <map />
  <node name="SystemIcons">
    <map>
      <entry key="WARNING"
      value="SystemIcons SECTION SHOULD NOT BE EDITED,
      IT IS READ ONLY" />
    </map>
    <node name="ICON_ID_10">
      <map>
        <entry key="icon.name" value="Shell Panel" />
        <entry key="icon.x" value="2" />
        <entry key="icon.y" value="34" />
        <entry key="icon.type"
         value="org.globus.cog.gridface.impl.
         desktop.icons.GenericIconImpl:System" />
        <entry key="icon.imageURI" value="images/32x32/co/terminal.png" />
        <entry key="App_Container_class"
         value="org.globus.cog.gridface.impl.shell.ShellPanelImpl" />
        <entry key="icon.launchStaticMain" value="false" />
      </map>
      <node name="App_Container_class_args">
        <map>
          <entry key="Arg0"
           value="org.globus.cog.gridface.interfaces.GridCommandManager" />
          <entry key="Value0" value="NOT_A_STRING" />
        </map>
      </node>
    </node>
    <node name="ICON_ID_11">
      <map>
        <entry key="icon.name" value="Java CoG Kit: Simple GridFTP Queue" />
        <entry key="icon.x" value="2" />
        <entry key="icon.y" value="66" />
        <entry key="icon.type"
         value="org.globus.cog.gridface.impl.
         desktop.icons.GenericIconImpl:System" />
        <entry key="icon.imageURI" value="images/32x32/co/fileshare.png" />
        <entry key="App_Container_class"
         value="org.globus.cog.gridface.impl.gftpanel.GridFTPPanelImpl" />
        <entry key="icon.launchStaticMain" value="false" />
      </map>
      <node name="App_Container_class_args">
        <map>
          <entry key="Arg0" value="java.lang.String" />
```

```
                <entry key="Value0" value="wiggum.mcs.anl.gov" />
                <entry key="Arg1" value="java.lang.String" />
                <entry key="Value1" value="2811" />
                <entry key="Arg2" value="java.lang.String" />
                <entry key="Value2" value="/home/pankaj/imageTest.jpg" />
                <entry key="Arg3" value="java.lang.String" />
                <entry key="Value3" value="wiggum.mcs.anl.gov" />
                <entry key="Arg4" value="java.lang.String" />
                <entry key="Value4" value="2811" />
                <entry key="Arg5" value="java.lang.String" />
                <entry key="Value5" value="/home/pankaj/test/imageTest.jpg" />
              </map>
            </node>
          </node>
        </node>
      </node>
    </node>
  </root>
</preferences>
```

NOTE: The icon count only reflects non system icons.

APPENDIX B

Desktop Command Line Usage

NAME

cog-desktop - Starts the Java CoG Kit Grid Desktop component

SYNOPIS

cog-desktop [-h] | [- f file] [-mxw integer] [-mxh integer]
[-ns] [-np] [-es]

DESCRIPTION

The Grid Desktop, also refered to as CoGTop, is a user centric grid workspace that functions and looks much like the popular Windows or KDE desktops. It bundles Grid functionalities already available in the Java CoG Kit into a single easy to use, persistent, and portable workspace that supports several desktop patterns like drag-n-drop. Additionally, we have added specialized desktop icons to represent RSL specifications that can be placed on to the Grid Desktop, and later droped on a job submission or a job specification icon to invoke a job submission on a service. All icons are able to edit their properties through their properties form.

The state of the Grid Desktop can be saved into an XML file. At restart it loads from its previous state.

Much like other desktop environments, the Grid Desktop also contains a toolbar, menu bars and internal frames.

OPTIONS

[(-help | -h)]
displays usage

[(-file | -f) <file>]
XML file containing Desktop Icon State.
NOTE: Will override any other desktop state file arguments

[(-maxwidth | -mxw) <integer>]

Maximum width for desktop. Used when starting desktop from different resolution monitor.

[(-maxheight | -mxh) <integer>]
Maximum height for desktop. Used when starting desktop from different resolution monitor.

[(-no-save | -ns)]
If present, the desktop will not prompt the user before closing. This is a handy debug flag for a quicker desktop exit.

[(-no-proxy | -np)]
Do not check for Grid Proxy on start up. Although, the user has the ability to regenerate proxy through the desktop.

[(-empty-state | -es)]
By pass loading from desktop file specified in .globus/desktop.properties, creating an empty desktop. System icons such as Shell and Directory Browser are still loaded.

EXAMPLES

1) To start the desktop from a previously saved state in file mydesktop.xml and ignore the default desktop file specified in .globus/desktop.properties, use the command

./cog-desktop -f mydesktop.xml

2) To load a desktop that is empty, does not check for proxy on startup and also does not confirm desktop exit, use the command

./cog-desktop -ns -np -es

3) To load a desktop with a specified resolution of 800 pixels wide and 600 pixels high.

./cog-desktop -maxwidth 800 -maxheight 600

TO DO

Add Grid Command Manager checkpoint information to the desktop

state file. Currently it is stored in a seperate file.

SEE ALSO

visual-grid-proxy-init, karajan-gui.txt

VITA

NAME

Pankaj R. Sahasrabudhe

DATE OF BIRTH

12/07/80

EDUCATION

1998-2003 Bachelors in Computer Science and Engineering at the University of Louisville. GPA: 3.7/4.0. Passing with Summa Cum Laude.

EXPERIENCE

6/14/2004 12/03/2004 Argonne National Laboratory (ANL), Argonne, IL
Graduate thesis research appointment.

8/14/2000 8/14/2004 University of Louisville, Louisville, KY
System administrator.

1/21/2002 5/03/2002 ExxonMobil Corporation, Houston, TX
Software Developer (Java, Visual Basic).

1/2001 5/2001 Structural Dynamics Research Corporation, Milford, OH
Software Developer (C, C++, CGI, Perl), Tester.

1/2000 5/2000 Structural Dynamics Research Corporation, Milford, OH
Software Tester.

ACTIVITIES

- Indian Students Association of Louisville –
  Founder, President, Web Master, 2000-2004.
- Association for Computing Machinery (ACM) – Vice-President, 2001.
- Department of Homeland Security,
  University of Louisville – member, 2004.
- Triangle Fraternity - Chapter Historian, 2001.
- Primerica Financial Services (member of Citigroup) -
  Independent Business, 2004.
- Student Ambassador for Speed School of Engineering,
  University of Louisville – 2003.

- Speed School of Engineering Student Council,
  University of Louisville – Public Relations Committee, 2003.
- Student Linux Users group (S+LUG),
  University of Louisville – 2000-present.
- National Public Radio (NPR) - member, 2004.

PROFESSIONAL MEMBERSHIPS

- Upsilon Pi Epsilon Computing and Information discipline honor society,
  2003-present.
- Tau Beta Pi honor society, 2001-present.
- Phi Eta Sigma honor society, 2000-present.
- Golden Key honor society, 2001-present.

HONORS

- Super Computing 2004 Best Poster Award,
  "Next Generation of the Java CoG Kit".
- Association for Computing Machinery (ACM),
  University of Louisville chapter, "Most Contributing Student" award, 2001.
- The National Dean's List, 1998-2000.