From Proposal to Production: Lessons Learned Developing the Computational Chemistry Grid Cyberinfrastructure

Rion Dooley
Center for Computation and Technology
Louisiana State University
Baton Rouge, LA 70803
dooley@cct.lsu.edu

Kent Milfeld
Texas Advanced Computing Center
Commons Center 1.154D,
J.J. Pickle Research Campus
10100 Burnet Road (R8700), Building 137
Austin, Texas 78758-4497
milfeld@tacc.utexas.edu

Chona Guiang
Texas Advanced Computing Center
Commons Center 1.154D,
J.J. Pickle Research Campus
10100 Burnet Road (R8700), Building 137
Austin, Texas 78758-4497
chona@tacc.utexas.edu

Sudhakar Pamidighantam
National Center for Supercomputing Applications
605 E. Springfield Ave.
Champaign, IL 61820
spamidig@NCSA.UIUC.EDU

Gabrielle Allen
Center for Computation and Technology
Louisiana State University
Baton Rouge, LA 70803
gallen@cct.lsu.edu

Abstract

The Computational Chemistry Grid (CCG) is a 3-year, National Middleware Initiative (NMI) program to develop *cyberinfrastructure* for the chemistry community. CCG is led by the University of Kentucky, and involves collaborating sites at Louisiana State University, Ohio Supercomputing Center, Texas Advanced Computing Center, and the National Center for Supercomputing Applications. In this paper we discuss our experience developing the CCG cyberinfrastructure in the first year of the project. We pay special attention to sociological as well as technical issues we faced, and look forward to challenges we foresee in the remaining two years.

1 Introduction

The term *cyberinfrastructure*, coined by an NSF Blue Ribbon Panel, refers to software which enables scientists to exploit cutting edge technology resources, including compute and data servers, visualization devices, instruments and networks, for advancing our research in science and engineering.

The Computational Chemistry Grid (CCG) [1] is a three year NSF funded project to develop a cyberinfrastructure to serve scientists engaged in studying molecular structure and function.

Computational chemistry algorithms and software are now widely used across a broad range of disciplines, including nanotechnology, biotechnology, and material science. Around the world users of both commercial and academic chemistry software packages, such as Gaussian, GAMESS, MolPro and NWChem, are major users of both high and middle end compute resources. CCG will leverage existing established Grid middleware to provide an easy-to-use integrated computing environment for these and other chemistry applications, including allocations on computing resources.

CCG is led by the University of Kentucky (UKy), and involves collaborating sites at Louisiana State University (LSU), Ohio Supercomputing Center (OSC), Texas Advanced Computing Center (TACC), and the National Center for Supercomputing Applications (NCSA). In this paper we discuss initial experiences developing the CCG cyberinfrastructure through the first year of the project and look ahead to challenges in the remaining two years. This paper discusses the important issues faced, both technological and sociological, as well as the chosen solutions.

The remainder of the paper is as follows. In Section 2 we give an overview of the CCG and the GridChem client application. In Section 3 we discuss the challenges of implementing the technological infrastructure and the roadmaps developed at the start of the project. In Section 4, we discuss some of the sociological issues of providing a community infrastructure. Finally, we outline the current state of the CCG and show how the community is adopting this new infrastructure as a means of facilitating their research.

2 Overview

The design of CCG, as shown in Figure 1, is a 3-tier architecture comprised of a client side GUI application, a middleware service, and a resource layer. The client application, called GridChem, is an open source Java application that remotely launches and monitors computational chemistry calculations on CCG supercomputers at remote sites. GridChem is a "lightweight" application and is distributed as an "executable jar" file. In the current release, installation includes downloading the client and installing a server certificate that enables secure communication with the GridChem Middleware Server (GMS) for authorization requests and notification. In future releases, we will leverage the Java Web Start technology to wave the certificate requirement and automate the process on behalf of the user

As mentioned in Section 1, the goal of GridChem is to create a powerful and useful tool for the computational chemistry community that allows users to easily submit, monitor, and manage their jobs using a large set of existing computational chemistry applications on a broad set of resources. User interface design is a very important aspect of GridChem and a significant amount of effort has been spent addressing chemistry and application specific issues relating to the user interface. GridChem provides additional features such as application-specific molecular editors, output file parsing, and "hooks" to leverage customized visualization tools. More information on the GridChem client can be found at [1].

It was realized very early in the project that a robust, Service-oriented Architecture (SoA) would be needed to gain widespread acceptance by the user community. Problems of scalability, distributed resource management, and the fluctuating nature of the Quality of Service (QoS) provided by each resource are inherent in a static grid implementation. This is evident by the grid community's growing adoption of SoAs, such as WS-Is Basic Profile Compliant Web Services [2] , and the upcoming Web Services Resource Framework (WS-RF) [3]. In both of these SoAs, services may

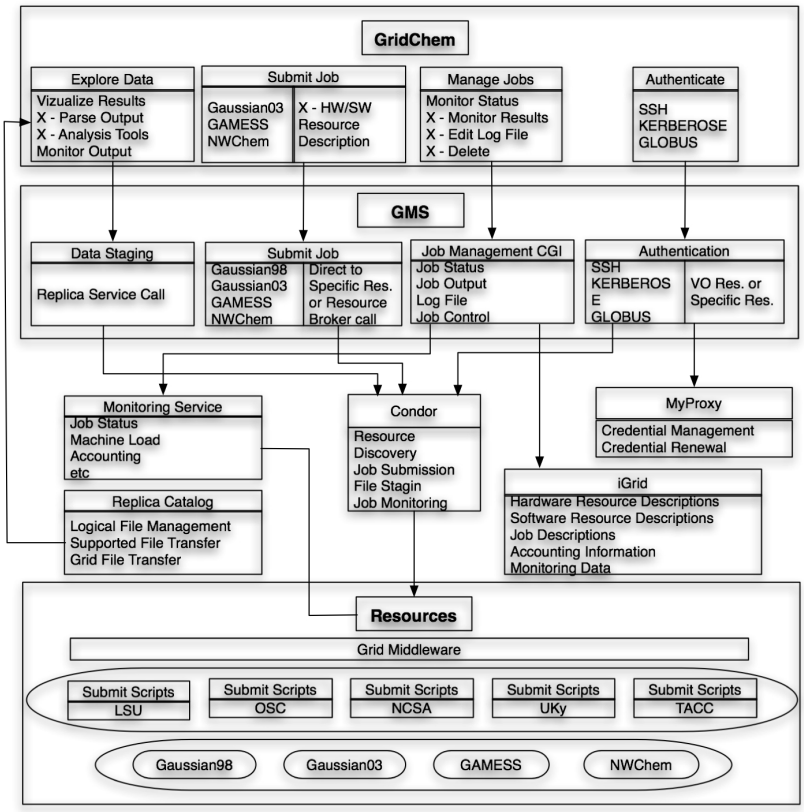

Figure 1: The planned architecture for the Computational Chemistry Grid. CCG is implementing a Service-oriented Architecture where the client utilizes the GMS for core functionality and the GMS in turn relies upon a series of grid and web services to provide functionality to the client. The current CCG architecture is very similar to the planned architecture. To move to this architecture, we will swap out existing CGI scripts for the GMS web services shown.

be composed hierarchically, allowing us to focus on developing the necessary meta-services needed for specific grid implementations, such as those in Section 3, rather than underlying low-level grid services. Using this approach, as the quality of the base services improves, so too will the quality of our meta-services.

Although the long-term goal of this project is to provide a robust, SoA, one of the primary deliverables in the first year of this project was to provide a production environment for users to submit, monitor, and retrieve output from jobs. To meet our short term goal of usability and to facilitate our long-term goal of implementing a robust grid architecture, we chose to first implement server-side functionality (ie. the middle layer of our architecture in Figure 1) in CGI scripts. Thus, the current middleware layer consists of basic grid middleware (Globus, NMI middleware distribution, etc.) and the client CGI scripts corresponding to individual client functionality. Using this approach, we can replace CGI scripts one-for-one with their corresponding web-service implementation as they become available. We choose web services to implement the future middleware layer instead of servlets or the existing CGI due to the complex nature of the final CCG architecture. As you will see in Section 3, the middleware must provide several complex features that would be difficult to achieve without heavy integration at the highest level. Using web services allows us

to integrate the accounting, job submission, security, and monitoring components at a level not possible in any of the individual underlying services.

The lowest level of the CCG Architecture is the resource layer which appears at the bottom of Figure 1. The resource layer consists of the physical resources, local schedulers, resource-specific low level information providers, and the software needed to run the computational chemistry applications on each machine. Included in this layer is a set of lexical scripts, called by the CCG middleware, that collect resource information and place each job into the local batch scheduling queue. The job queueing scripts are application specific and tailored to each machine's unique characteristics. As the middleware services mature, the scripts will be phased out in favor of server-side resource brokering that will leverage robust software information provisioning.

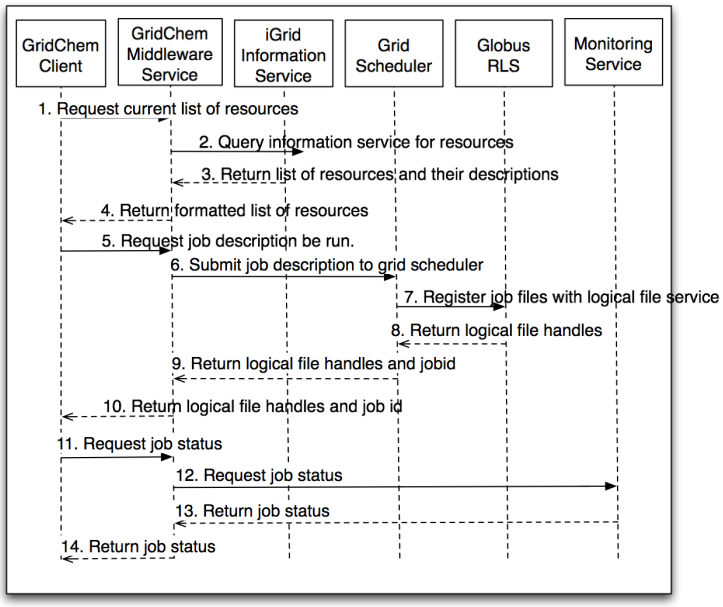

Figure 2: Sequence diagram of GridChem interaction with the GMS.

Further information on the interaction between the different CCG architectural layers is shown in Figure 2. Notice that each action in the client is reflected by a call to the corresponding GridChem Middleware Service. The GMS in turn, will leverage one or more underlying services to perform the requested action. In this manner, GMS takes advantage of continually maturing grid services without requiring significant development itself. More information on the CCG Architecture can be found at [4].

3 Technological Issues

Creating a production grid environment poses several significant technological problems related to security, software selection, account management, resource brokering, and accurate and up-to-date information provisioning. In this section we address each of these issues in turn.

3.1 Security

The CCG currently supports GSI, Kerberos, and Secure Shell security mechanisms. The decision to support these various mechanisms was a trade-off between wanting to develop a full grid architecture, and needing to gain wide community acceptance. Initially we decided to support all mechanisms and then gradually migrate users towards GSI security via our community account. The CCG community account is a multi-site allocation provided by each participating site to the community. Through the community account, we provide full GridChem functionality to users without requiring them to have an allocation on any individual machine.

Authentication and authorization both require significant planning. We first need to authenticate the user in order to map their (potentially numerous) userids to their unique GridChem account. This situation occurs when a user has multiple usernames on a machine and possesses more than one grid certificate that can authenticate. In addition to keeping track of the user's multiple identities, we must keep track of the authorization technique to use with each identity (ie. GSI, Kerberos, Secure Shell). Without such mechanisms, it is not possible to track account usage and determine the correct way to authorize the users on each machine.

Because we knew many users would not be familiar with grid technology, we could not require them to use their grid certificate for each operation they wished to perform. Additionally, we could not ask them to install a large, complicated suite of grid middleware simply to use the GridChem client. To avoid these issues, we allow users to authenticate to the GMS and perform all grid functionality, such as creating, pushing, and pulling a user's credentials from MyProxy [5], there, where the installation of middleware and details of executing grid functionality are handled on the user's behalf.

An example authentication session using the GSI mechanism is as follows. A user starts the GridChem client and opens the "Authenticate" panel. There, they select "Myproxy" authentication, enter their CCG username and password, then click on the "Login" button. GridChem then encrypts the username and password and sends them as arguments to the GMS Authentication Service. The service checks to see if the user's information is correct, then pulls a community credential from our Myproxy server. Upon completion of this step, the user is authenticated and can use the full functionality of the client without thought for the underlying security mechanisms or having to authenticate again.

3.2 Accounting

Accounting is a fundamental concern for this project. Because we provide access to a community account, we must track not only overall account usage, but also individual user utilization. Further, we must solve each security scenario separately due to the unique characteristics of the job submission workflow related to each technique. For the grid scenario, we solved this problem by mandating individual user registration before granting use of the GridChem client application. By controlling all account allocations, when a user logs in, we can track their activity by mapping their chosen authentication technique to their CCG account using the DN from their grid certificate. We currently store this information in a server-side database. By doing this, we can manage individual accounts via a web interface or through GMS.

A problem which has proven to be much more difficult and remains an open question for this project is accounting for user activity when the user submits jobs outside of GridChem. One solution is to

rely on regular parsing of the history records from the local schedulers and reliable queue monitoring. Often times, however this information is not readily available. Even assuming this information is available, we still then need to develop mechanisms to map local machine accounts to CCG users. When using GMS, we can locate user jobs by querying our database for using their unique user id. In more general cases, however, we do not have such a luxury. Reverse mapping is very difficult, if not impossible considering the user may use a special project account, a community account, or simply a personal account issued under a different site allocation to run jobs not affiliated with CCG.

To date, we have not found a definitive solution to this problem. Our primary focus at this time is verifying accurate accounting of GridChem jobs. Once we are able to guarantee that all jobs are audited correctly, we plan on incorporating information on the user's other jobs into the auditing process. However, as we mentioned above, it may not be possible to account for jobs not affiliated with GridChem. In practice, this issue may not prove to be a problem. Users may use GridChem for all their computational chemistry needs, or not hold GridChem liable for tracking their outside jobs. Without an active user community to examine the situation further, we cannot be certain. We expect that the general case will remain an open question until the project is well into its production phase.

3.3 Resource Brokering

As mentioned in Section 2, the problem we address with the CCG is to provide cyberinfrastructure for the computational chemistry community that enables them to submit and manage jobs using a select set of well-known applications. Narrowing our focus from the general case to this specific case of only supporting a few known applications simplifies the task of resource brokering. We know what applications our user community will employ, we know the finite list of dedicated machines on which the user will run these applications, and we are able to control the means in which this introduction will happen. Using this information greatly reduces the complexity of our task and allows us to again leverage the SoA paradigm to perform resource brokering at two distinct levels.

At the lowest level, we use grid schedulers to submit and manipulate the user's job on every resource. Existing tools such as Condor [6], the Grid Resource Management System (GRMS) [7], and GRAM [8][9] fit this description. In order to avoid dependence on any one scheduling service, we leverage the Grid Application Toolkit (GAT) API [10] which allows us to interchange grid schedulers without altering our code base. It also allows us to take advantage of the best features of each technology and provide an overall service that is more sophisticated than it's individual parts. A good example where we have the opportunity to leverage existing technology to solve low-level problems is proxy certificate management. Computational chemistry jobs can vary in length from a couple hours to many months. As job run time increases, so to does the possibility that a user's proxy will expire long before their job finishes. If the user's proxy expires, all grid-based output file transfer will fail due to an expired credential. Condor-G is a grid scheduler that performs credential management on behalf of the user. Thus in the case of long running jobs, as an alternative to generating a credential with an extremely long life, we are able to use Condor-G as the underlying grid scheduler to renew the user's proxy credential on their behalf.

At the highest level, we will provide sophisticated services to the user such as throughput scheduling, economic scheduling, job monitoring, and notification. Intelligent scheduling of jobs, using different criteria for optimality, is one of the second year goals of the project and is crucial in ensuring efficient

use of grid resources. By definition, throughput scheduling seeks to maximize job throughput by minimizing job turnaround time. Aside from requiring dynamic information that reflects current resource utilization, throughput scheduling necessitates reasonable values of three parameters that determine total job execution time, namely queue wait, data transfer and application run times. To obtain estimates of these parameters (within some specified error bounds), our metascheduler will utilize a web services-based prediction toolkit that implements the instance-based learning (IBL) method pioneered by Smith [11].

The job scheduling, monitoring and notification services are made available through the rich resource descriptions pushed to our information service. Detailed software and hardware resource descriptions allow us to integrate accounting into the decision making process, which in turn, enables better decisions than could be made by a third-party broker.

With better information comes better accounting, with better accounting comes better brokering. As we mentioned in section 3.2, we are working to provide more mature information providers and an advanced accounting system. As of this writing, these systems are not in place, thus our current resource brokering capabilities are dependent on the strength of the underlying grid schedulers we employ: currently Condor-G and GRAM.

3.4 Information Provisioning

Accurate and dependable information provisioning is the largest single challenge of this project. Without reliable information from all aspects of the system, we cannot make the necessary and intelligent decisions on the user's behalf. In line with our SoA architecture, we adopted the iGrid information service [12] to supply resource information to the CCG. iGrid is a hierarchical information service shown to perform several times faster than the Globus MDS [13]. The basic installation of iGrid, however, did not provide resource descriptions that were descriptive enough to meet our needs. Fortunately, iGrid is extendable and by working closely with the iGrid development team we were able to create a resource description schema suitable for the CCG. This proved to be relatively straightforward, taking less than a week, however, creating providers to discover local resource information took significantly longer.

Over the last month we have created the Job And Machine Monitor Service (JAMMS). JAMMS is a Perl script that pulls information on queues, running jobs, processor and machine utilization, wait time, and history. JAMMS is run as a cron job at each site. By default, the script is set to run every 5 minutes as a local user (e.g. ccguser). The Perl script uses the Perl Database Interface (DBI) module for sending information to a MySQL data base (at www.gridchem.org). A PHP program is used to extract information from the database and present it to the user, through their browser.

JAMMS programs, called filters, execute batch utilities (for either PBS, LSF, or LoadLeveler), and extract (filter out) needed information. Two or three batch utilities might be invoked from within the Perl script to obtain the relevant information. For LSF, a single API program was developed to quickly extract all relevant information. We are currently in the process of integrating the iGrid API into JAMMS so it can serve as a provider to the iGrid information service.

4 Sociological Issues

Equally as difficult as the technological problems associated with building a production grid are the sociological issues. Accommodating individual site policies, putting software in place, administering the various resources in a consistent way, and finding ways to to avoid stepping on each others' toes are all obstacles to overcome before a grid can ever be deemed, "production"-worhty. This section discusses how we have, in the past, and continue to address each of these issues.

4.1 Community Trust

Participating in a grid can prove difficult because it often involves compromise on security and policy issues. Often this means agreeing to place a certain degree of control over the way local systems are run in the hands of someone else. It is an issue of trust. We know our colleagues, but how well do we really know them? Do we trust them enough to blindly issue accounts? Do we trust them enough to commit time and manpower to install and maintain a predefined, and often changing, software stack? Do we trust them to play by the rules by which we play? In short, how can we make sure that we're good grid participants without leaving ourselves exposed?

In the CCG, we approach each of these issues with a healthy dose of open communication. We are fortunate enough to represent sites of sufficient size and manpower to carry out the project, yet comprise a group of people small enough to develop interpersonal relationships. Weekly Access Grid [14] meetings allow us to "see" each other on a regular basis and connect visually with the faces and voices behind the words. We implemented a solid system of documentation and discussion using several media from the telephone to email and from instant messaging to group Wiki pages [15]. This redundant level of communication and familiarity reduces the amount of insecurity and hesitation that exists when "strangers" are forced to work together.

Complementary to building trust relationships, we must build a level of certainty that people are capable of performing the task at hand and dependable enough to carry out their work to the last detail. Sometimes that means saying, "I won't have time to do that." or, "I'm not sure how to do that, let me do some reading and get back to you." Being honest and accurate in our estimates and asking for help when we necessary has allowed us to more effectively work as a team.

As is true in life, with trust and certainty come confidence in the integrity of our colleagues. Through positive experiences with each other, we are reinforced with the knowledge that we are working with a group of people "just like us." In short, by committing to be good grid participants ourselves, we create an community of people who want to be good participants themselves.

Of course, not everything always works out as planned. We all experience setbacks, miss deadlines, and inevitably have our own "best ways" of doing things. The important thing in our experience has been a willingness to listen and think things through. We cannot afford to let our egos get in the way of the project at hand. Individually we all win some and lose some, but in the end the goal is to make the project successful. If that happens, then all of us are winners.

4.2 Proprietary Licensing

Not everyone supports the concept of "sharing" resources. There are significant obstacles to overcome, both legal and historical, before industry accepts the notion that a site license means un-

limited use at that site. Unique pricing, wildly varying negotiation experiences, and rejection were all situations CCG member sites experienced when trying to purchase the license required to run different Computation Chemistry applications. In the end we decided that not every site would provide a complete set of end user applications. We believe that, for our needs, it is sufficient that each applications is available at a subset of CCG sites. This solution still gives the users several options where they can run their jobs and allow sites who cannot negotiate acceptable contracts with software vendors to participate in the project.

5 Community Acceptance

Having already released the first version of our software at the GridChem Workshop in April, we are now benefiting from user feedback and input from the computational chemistry community. Through surveys given at the workshop and conversations with individual users, we found that many users were very enthusiastic about the notion of a community account for running their jobs. The SSH authentication interface was extremely successful and users stated that they would be willing to adopt using grid certificates through a similar interface - especially if it meant gaining access to additional compute time.

One recurring request from many users was incorporation of a workflow engine into the GridChem client. Several researchers are currently performing task farming and/or more complex jobs that GridChem could potentially support. Because this is a valuable feature driven by user requirements, we will incorporate this into GridChem, however not in the near term as this functionality requires adaptation at both the client and server levels and would be best incorporated at a later stage in the project.

The last request users made was for more user-friendly file management. Keeping track of user output files is not an easy problem. It boils down to the same issues we face with accounting. As a first step, we are integrating Trebuchet, developed as a part of the Open Grid Computing Environment (OGCE) project [16], to provide a GUI-based grid file browser for the users. This feature, however, will not be available until the next full release.

We would like to thank Ian Kelley and Jon MacLaren for thoughtful review, as well as Michael Sheetz and the UKy development team for their contribution to the GridChem GUI. This work was funded in part by the National Science Foundation, Award #0438312.

References

[1] GridChem Project WebSite
http://www.gridchem.org/

[2] Web Services Interoperability Organization Basic Profile Version 1.1, Final Material, August 2004. http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html

[3] The Web Services Resource Framework. http://www.globus.org/wsrf/

[4] K. Milfeld, C. Guiang, S. Pamidighantam, J. Giuliani. Cluster Computing through an Application-oriented Computational Chemistry Grid. Proceedings of the 2005 Linux Clusters: The HPC Revolution.

[5] Myproxy WebPage
http://myproxy.org

[6] D. Thain, T. Tannenbaum, and M. Livny. "*Condor and the Grid*", in F. Berman, A. J. G. Hey, G. Fox, editors, *Grid Computing: Making The Global Infrastructure a Reality*, John Wiley, 2003.

[7] GridLab Project Website
http://www.gridlab.org/

[8] Globus WebPage - GRAM Overview.
http://www-unix.globus.org/toolkit/docs/3.2/gram/key/index.html.

[9] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A Resource Management Architecture for Metacomputing Systems. In D. Feitelson and L. Rudolph, editors, Job Scheduling Strategies for Parallel Processing (Proceedings of the Fourth International JSSPP Workshop; LNCS #1459), pages 6282. Springer-Verlag, 1998.

[10] G. Allen, K. Davis, T. Dramlitsch, T. Goodale, I. Kelley, G. Lanfermann, J. Novotny, T. Radke, K. Rasul, M. Russell, E. Seidel, O. Wehrens. "*The GridLab Grid Application Toolkit.*" HPDC 2002: 411.

[11] W. Smith. "Improving Resource Selection and Scheduling using Predictions," in Grid Resource Management. J. Nabrzyski, J.M. Schopf, J. Weglarz (Eds). Kluwer Publishing, Fall 2003.

[12] iGrid Website.
http://www.gridlab.org/WorkPackages/wp-10.

[13] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, S. Tuecke. "*A Directory Service for Configuring High-Performance Distributed Computations*". Proceedings of the 6th IEEE Symposium on High-Performance Distributed Computing, pp. 365-375, 1997.

[14] Access Grid WebPage
http://www.accessgrid.org/agdp/

[15] Wiki WebPage
http://wiki.org/wiki.cgi?WhatIsWiki

[16] OGCD Website.
http://www.collab-ogce.org/nmi/index.jsp.