# Towards a Top-Down Approach to Teaching an Undergraduate Grid Computing Course

Barry Wilkinson
Department of Computer Science
University of North Carolina Charlotte
9201 University City Blvd.
Charlotte, NC 28223 USA
abw@uncc.edu

Clayton Ferner
Department of Computer Science
University of North Carolina Wilmington
601 S. College Rd.
Wilmington, NC 28409 USA
cferner@uncw.edu

## ABSTRACT

Early undergraduate Grid computing courses generally took a bottom-up approach to Grid computing education starting with network protocols, client-server concepts, creating Web and Grid services, and then progressing through the underlying Grid computing middleware, security mechanisms and job submission all using a Linux command-line interface. We describe a new approach to teaching Grid computing beginning with a production-style Grid portal, registration process, and job submission, and then leading into infrastructure details. We incorporate seven assignments, several of which require students to install Grid computing software on their own computer or lab computers rather than using centralized servers. Students complete a "capstone" mini-project. Typical projects included creating a JSR 168 portlet user interface for an application. We describe our experiences using this new course structure.

## Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *client/server, distributed applications*

## General Terms

Design, Experimentation, Security, Human Factors, Standardization.

## Keywords

Grid Computing, Globus, Grid Portal, Undergraduate Education.

## 1. INTRODUCTION

Grid computing takes advantage of the Internet by using geographically distributed computers for collaborative problem solving. In Grid computing, different organizations can supply resources and personnel, and the Grid infrastructure can cross organizational boundaries. Grid computing has become an important concept for high performance computing. This concept has many benefits including solving problems that could not be solved previously because of limited computing resources (e.g. searching for new drugs). Grid computing has found its way into the permanent Computer Science curriculum at many schools in the country.

Grid computing entered into Computer Science programs originally as graduate-level topics courses within a single department. Subsequently, undergraduate Grid computing courses were developed. In 2004, we developed an undergraduate Grid computing course that crosses organizational boundaries using resources at several North Carolina universities. The course was broadcast across North Carolina using the televideo facilities of the North Carolina Research and Education Network. Fourteen universities and colleges participated included minority-serving universities, state universities, and private colleges. The course was first taught in Fall 2004 and again in Fall 2005, and is described fully in [2,11,12]. We now describe an extensively revised version of the course with a more top-down approach. The course now starts with the use of a Grid computing portal, leading through details of Grid computing infrastructure with seven hands-on assignments, finally cumulating in small group mini-projects. The revised course was taught in Spring 2007.

Computer Science courses are often taught with a top-down approach. This means that the initial viewpoint is from an overview perspective. As the semester progresses, the lessons investigate various aspects of the course in more detail. The main advantage of this approach is that students have a better understanding of where the course is headed and how the different parts fit together. However, Grid computing has so far been taught with a bottom-up approach. Students are first exposed to low-level details of working with a Grid, such as installing or configuring middleware, creating user credentials, executing simple commands, or creating trivial Web or Grid services. Later in the semester, students will learn about the higher-level concepts such as file staging, using a portal or workflow editor, high performance computing, or combining various features of a Grid into a more complex solution.

We redesigned our course to give students a more top-down perspective. The design might best be described as alternating between a high-level view and a low-level detail oriented view. Students still need to be exposed to low-level details of Grid computing fairly early in the semester to give time for them to deal with more complex concepts. However, we feel that the new design gives students a new perspective that puts all lessons into a cohesive context. In this paper, we discuss the design of the course and our efforts to change the perspective to a top-down approach.

## 2. RELATED WORK

One of the first funded projects for teaching Grid computing at the undergraduate level was an NSF Course, Curriculum, and Laboratory Improvement (CCLI) project awarded to Bina Ramamurthy and Bharat Jayaraman in 2003 [6]. This was a collaborative project between SUNY at Buffalo and SUNY College at Geneseo. The project resulted in courses being offered at the advanced undergraduate level from a distributed computer system perspective beginning with RMI (remote method invocation). In 2004, NSF funded two further CCLI projects on undergraduate Grid computing education. In one project, investigators at the University of Arkansas (Amy Apon) and at Lewis and Clark College (Jens Mache) developed a collaborative undergraduate Grid computing course [7]. In the other project, investigators at UNC-Charlotte (Barry Wilkinson), Western Carolina University (Mark Holliday), and UNC-Wilmington (Clayton. Ferner) developed a collaborative undergraduate Grid computing course [8]. We describe in this paper a continuation of that project. All previous projects concentrated upon command-line interfaces and distributed computing techniques [1, 3, 4, 5].

## 3. COURSE DETAILS

### 3.1 Student Resources

The course home page, located at http://www.cs.uncc.edu/~abw/ITCS4146S07/ provided access to everything the students needed, including more than 800 PowerPoint lecture slides, reading list, assignment instructions, WebCT for assignment submission and multiple-choice quizzes, course Grid portal, and streaming video recordings of classes available to students during the semester. Figure 1 shows a sample of the streaming video with the three universities that participated in the course experiment (UNC-Asheville, UNC-Charlotte, and UNC-Wilmington). Previously, we had many universities participating in the course, but because of the major changes we made, we limited participation.

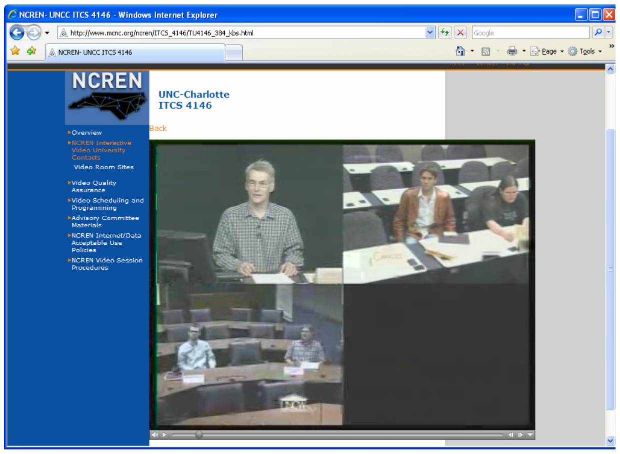
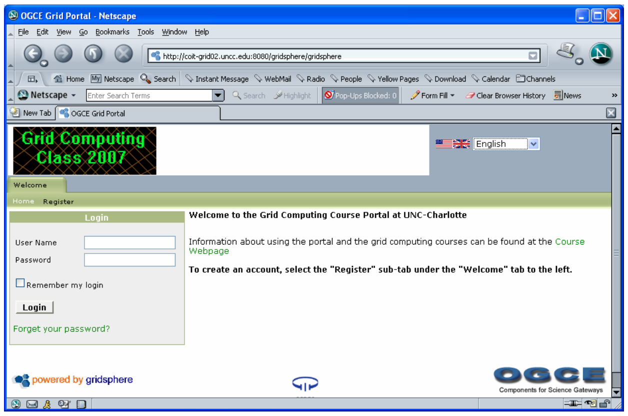

NCREN- UNCC ITCS 4146 - Windows Internet Explorer
NCREN
UNC-Charlotte
ITCS 4146
Back
Overview
NCREN Interactive Video University Contacts
Video Room Sites
Video Quality Assurance
Video Scheduling and Programming
Advisory Committee Materials
NCREN Internet/Data Acceptable Use Policies
NCREN Video Session Procedures

**Figure 1. NCREN Streaming video of class**

Some new and revised assignments called for students to install complex Grid computing software on their own computer (or lab computers) rather than use the main Grid computing systems. Past experience with students using a central system for software development was fraught with problems, as we shall describe later. The software is available for Windows, Linux, and Mac although all the students in the class of Spring 2007 used Windows systems. Using personal or lab computers was one of the successes of the redevelopment. Of course, we had to establish at the beginning that everyone had access to a computer.

### 3.2 Assignment 1: Using a Grid Portal

We started the course with an early assignment in the first class even before getting into the details of Grid computing, which asked students to register themselves as users using a Grid portal. Our Grid portal is based upon the Gridsphere toolkit [9], a widely used portal toolkit in production Grids. A Grid portal provides single sign-on to all Grid resources, that is, once logged in with a password, access can be made to reach local and distant resources without having to supply passwords subsequently. The course Grid portal is shown in Figure 2.


OGCE Grid Portal - Netscape
File Edit View Go Bookmarks Tools Window Help
Home Netscape Search Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Netscape Enter Search Terms Search Highlight Pop-Ups Blocked: 0 Form Fill Clear Browser History News
New Tab OGCE Grid Portal
Grid Computing Class 2007
English
Welcome
Home Register
Login
User Name
Password
Remember my login
Login
Forget your password?
powered by gridsphere
OGCE
Components for Science Gateways

http://coit-grid02.uncc.edu:8080/gridsphere/gridsphere

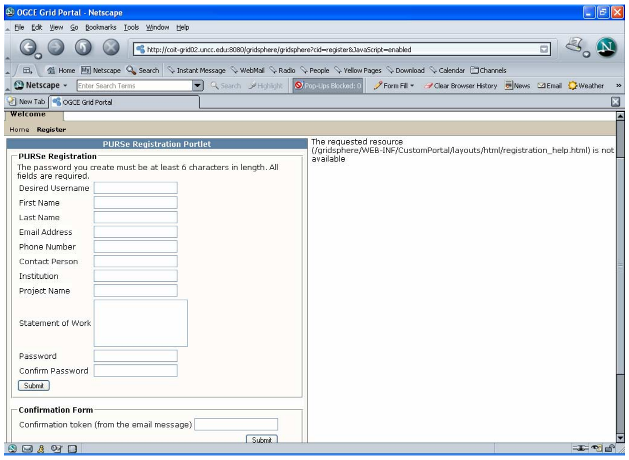
Welcome to the Grid Computing Course Portal at UNC-Charlotte

Information about using the portal and the grid computing courses can be found at the Course Webpage

To create an account, select the "Register" sub-tab under the "Welcome" tab to the left.

**Figure 2. Main portal login page**

We installed the PURSe registration portlet [10], which greatly simplifies account set-up. Students select the registration tab on the portal main page, which leads them to the PURSe registration form shown in Figure 3. They then enter details and request an account.


OGCE Grid Portal - Netscape
File Edit View Go Bookmarks Tools Window Help
Home Netscape Search Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Netscape Enter Search Terms Search Highlight Pop-Ups Blocked: 0 Form Fill Clear Browser History News Email Weather
New Tab OGCE Grid Portal
Welcome
Home Register
PURSe Registration Portlet
Submit
Submit

http://coit-grid02.uncc.edu:8080/gridsphere/gridsphere?cid=register&JavaScript=enabled

PURSe Registration

The password you create must be at least 6 characters in length. All fields are required.

Desired Username
First Name
Last Name
Email Address
Phone Number
Contact Person
Institution
Project Name
Statement of Work
Password
Confirm Password

Confirmation Form

Confirmation token (from the email message)

The requested resource (/gridsphere/WEB-INF/CustomPortal/layouts/html/registration_help.html) is not available

**Figure 3. Registration portlet**

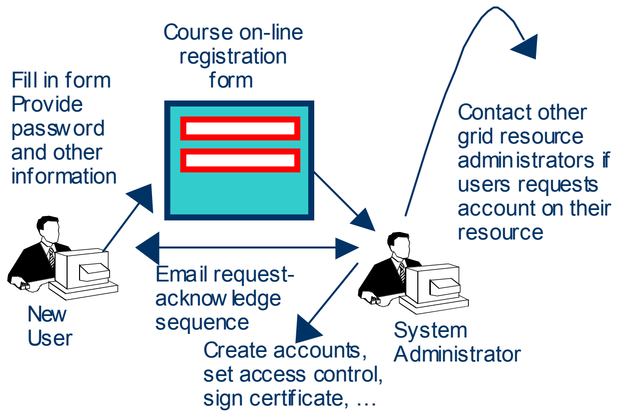

Course on-line registration form
Fill in form Provide password and other information
Contact other grid resource administrators if users requests account on their resource
Email request-acknowledge sequence
New User
System Administrator
Create accounts, set access control, sign certificate, …

**Figure 4 Registration process**

Behind the scenes, a number of activities occur as illustrated in Figure 4. These activities are somewhat hidden from the students at this stage. Student will not understand the creation of credentials (private key and certificate request). Although the students did not yet understand what is happening for them, they are becoming oriented to the Grid through the portal interface. It was important for the students to do these first few steps immediately so that we could get certificates signed and issued and they could proceed with the rest of the assignment.

After certificates were signed and placed on the MyProxy server, students could complete the assignment, again using the portal, by getting a proxy and submitting a simple job to execute on the Grid. They test the system with a prewritten job (Linux echo command). They then write a simple program in Java (factorial program), compile it to a class file on their own or a lab computer, which is then uploaded into the server for execution using the GridFTP portlet within the portal. Hence, students get experience in using the portal to do some basic tasks.

While the students are working on this first assignment, we delivered the lectures on an overview of Grid computing, including topics such as Grid computing concepts, virtual organizations, computational Grid projects, Grid computing networks, Grid computing infrastructure, and software components. Up to this point, the students have been introduced to the concept of a Grid as a computing resource. They know that there are machines they can put to use to accomplish some task. How that happens was still a mystery to them.

## 3.3 Assignment 2: Using the Grid Through a Command Line

In the second assignment, students accomplish the same tasks as in Assignment 1, except that everything is done through the command line. Students must create a certificate request, submit it to the Certificate Authority, create a job submission file, and submit the job to the Globus Grid Resource Allocation and Management (GRAM). The assignment has not changed significantly from previous offerings of the course. However, we still include the assignment to contrast job submission from a command line interface against a portal interface as well as to give students a perspective of what the portal is doing for them under the surface.

In the lectures at this point, we cover the command line interface approach to Grid computing as well as delivered lectures on XML, Job Management, Web Services, and Security.

## 3.4 Assignment 3: Using a Scheduler to Submit a Job

Next in the course are lectures on schedulers and an assignment using a scheduler such as Condor or Sun Grid Engine. We used Condor in Fall 2004 and Sun Grid Engine in Fall 2005. Sun Grid Engine required a third-party adapter for the Globus Grid computing software but has a nice GUI interface. For Spring 2007, we returned to Condor as it provides interesting features for job matching through ClassAd and workflow though DAGMan and is widely used as a cluster scheduler. There is also a Grid version of Condor called Condor-G.

In the assignment, students are asked to create a job submission file and submit a job to the Condor scheduler. It was our intention that the students be able to submit jobs to the Globus container through Condor as well as to Condor through Globus, giving them a view of local versus meta-scheduling, but this was left for a future offering of the course together with true meta-scheduling. We did however have a guest speaker later in the course describe real-life meta-scheduling using Gridway.

## 3.5 Assignment 4: Installing the GT4 Core and testing a simple GT4 Grid service

The next part of the course is a return to the low-level details of creating a Grid service, which is a key aspect of recent of Grid middleware. We delivered lectures on concepts such as Globus 4.0 Grid services, using Web services for Grid computing, stateful Web services, Grid computing standards, Open Grid Services Architecture (OGSA), Web Services Resource Framework (WSRF), programming GT 4.0 Grid services, and the GT 4.0 container. Students are given an assignment to create, build, and deploy a Grid service, then use the Grid service by creating, building, and using a client.

In previous offerings of the course, this assignment was done earlier in the course and required students to use one of the servers at their institution. However, this did not work out well. The original Grid service assignment required students to create their own Grid service, which requires a Grid service container host. Using one container (or multiple containers operating on different ports) requires all students' services to be uniquely named, which was done previously by running a script that renamed student services. When a student needed to modify and re-deploy the service, it was necessary to restart the container to see the service. Students would do this many times and re-deployment would affect all students using the same container. It was easily possible to overload or crash the system. Asking students to do the service development on their personal computers eliminated such problems. However, students needed to install the Globus core and associated software to do this work.

The software required for Spring 2007 was JDK 1.4.2+, Ant 1.5.1+, Python 2.4+, and Globus 4.0 core. Installing the software is quite easy to do although many students had not downloaded and installed such software before. The software required more low-level work than most Windows software they had installed. They had to obtain packages from different sources and set environment variables and paths. Once installed, students needed to issue commands at a Windows command prompt.

We did provide a version of the assignment that used a server (basically the 2004/2005 assignment) as a backup plan for those students that might not be able to install the software, but all of the students managed to install the software on Windows computers. (None of the students in the Spring 2007 class used Linux or a Mac although they could have done so.)

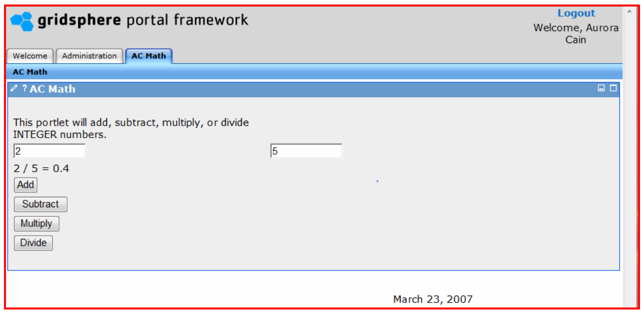
## 3.6 Assignment 5: Using GridNexus to Create Workflows that use Web and Grid Services

We returned to user interfaces in the next phase of the course and gave the students an assignment where they have to interact with Web services and submit jobs using a workflow editor. In this assignment, the students are asked to install the GridNexus workflow editor software and create workflows that would use Web and Grid services. Although the students used personal computers for GridNexus, the software functions as the client to services on a Grid machine. We wanted to have students use GridNexus to interact with the Grid service that they themselves built and deployed but this would be difficult for students who were not successful with Assignment 4. It is desirable to have each assignment be as self-contained as possible. Therefore, we provided the services the students could use in this assignment.

After configuring GridNexus to work with the services, students are asked to create workflows that created service instances, make use of them to accomplish some task, and then destroy the service instances. Students are also asked to submit a job to run on a remote machine using the workflow editor through the Globus GRAM. They could then contrast this job submission with using a Portal (Assignment 1), using a command line (Assignment 2), and using a scheduler (Assignment 3). Similar to a Portal, GridNexus will hide many of the low-level steps necessary to submit a job such as the construction of the job submission file.

## 3.7 Assignment 6: Portlet Design

This was a completely new assignment for Spring 2007 in keeping with introducing a higher-level interface to students. It also required students to use their own computer. Students are asked to create, build, deploy, and use a portlet inside a portal container that they installed. It is necessary for them to install GridSphere and other software (such as ANT and Tomcat). After successfully installing and running the container and GridSphere, students then had to create a prewritten JSR 168 portlet, build it, deploy it in GridSphere, and then code their own portlet. Figure 5 shows a sample student portlet as a front-end to a simple math program.



This portlet will add, subtract, multiply, or divide INTEGER numbers.

2 | 5

2 / 5 = 0.4

March 23, 2007

**Figure 5. Student Math Portlet**

This may not look difficult, but it requires student to understand, code, and deploy a JSR 168 portlet. They have to write the underlying Java math program, which calls a JSP file (Java Server Pages) and modify several deployment descriptor XML files, get it to work, and write up the work in a report, all in a week or so (including installing the software).

## 3.8 Assignment 7: MPI Assignment

At this stage in the course, we move on to parallel computing using MPI and high performance computing. Students are given an assignment to create a parallel program using MPI, build it, and submit it for execution on the Grid. In Assignment 7, students are asked to create MPI programs to implement matrix multiplication and dot product. (Students could install MPI on their own computer for code development if they wished although we are not aware of any who did.) We were only able to spend a few lectures on parallel computation, so we could not ask them to implement anything more complicated. Furthermore, the communication latencies between geographically distant processors are too large to make it feasible to employ parallel computation that requires a significant amount of communication. Applications that are suitable for parallel execution on a Grid include such things as "embarrassingly parallel" applications and Monte Carlo applications. However, an assignment such as this one only introduces the students to the possibilities that parallel execution promises as well as the elusiveness of speedup.

## 3.9 Mini-project

The mini-project was also new for Spring 2007. Students worked in teams of three students. Each team is given the objective of creating a new Grid computing assignment that builds upon the previous assignments and concepts learned in the class, and produce an assignment write-up with the solution. The assignment should involve the creation of a Grid computing application combined with a graphical interface such as a workflow editor or a portal. Note the requirement of a high-level user interface. We provided students with a list of possible mini-project ideas, but they could design their own assignment. Whichever they chose, they had to present a single page write-up describing the idea for our approval by a published date, after which they could start in earnest. Students are asked to design the work as a class assignment that could be given to their classmates. We did this so that we might use their mini-projects as actual assignments in future classes. Furthermore, this requirement would give students a way to focus on the dissemination of the project. Finally, student teams had to prepare a report written as a class assignment and present to the class a 10-minute presentation of what they did, what they accomplished, and what they learned.

While students were working on their mini-projects, we invited three expert guests who are working on real-world Grid projects to give presentations on those projects. These presentations gave students an appreciation that Grids are more than just experimental systems.

## 4. EXPERIENCES

Our experiences have been very positive. It does require immense work to prepare for a hands-on Grid computing course. It is critical that all assignments are fully tested prior to the start of class and that all computer systems are reliable and the software maintained. The assignments went much smoother by requiring students to use personal computers when possible. Students responded positively to using their own computers that were under their direct control. Some students did have problems with their installations but nothing that could not resolved quickly such as not setting paths or using the wrong version of the software. In one case, using a most recent version of JDK (1.6 instead of the versions it was tested on: JDK 1.4.2 and JDK 1.5) turned out to be incompatible. Certainly some Grid computing assignments still have to be done on a Grid platform, such as submitting and scheduling jobs, but using personal computers where possible simplifies code development. Thankfully, we did not experience any students causing catastrophic system problems with installing the software on their personal computer although we warned students to checkpoint their system prior to installation. Most lab machines were installed with Deep Freeze that insured students would not cause them to be inoperable. The only downside of using Deep Freeze is that the students had to complete their assignment in one sitting or repeat the installation.

In Spring 2007, we had increased the number of assignments from four to seven and added a mini-project so it was critical that no one got behind with assignments in the 15-week semester. Each assignment was allocated 1-2 weeks to complete (3 weeks for mini-project). As with the Fall 2005, we posted three dates for each assignment, a date the assignment was set, a date that students had to report any system problems that were preventing them from proceeding with the assignment (roughly 3-4 days after the date set) and a date the assignment was to be submitted. The date to report system problems required students to start the assignment immediately. It turned out that in Spring 2007, no one reported systems problems and each assignment could move forward on schedule.

## 5. CONCLUSIONS

In this paper, we describe a new version of a geographically distributed undergraduate Grid computing course, which uses a more top-down approach, and starts with production-style Grid computing portal. We also introduced JSR 168 portlet design and a mini-project that brought together the materials in the course. Our new course calls for students to install some complex software on their own computer or lab computers in order to do some assignments. This approach significantly reduces the problems of a class attempting Grid computing work on a centralized server, especially Grid service code development.

## 6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0533334. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We wish to thank our teaching assistants at UNC-Charlotte, Jeremy Villalobos and Jasper Land who were responsible for maintaining the Grid platform and assisted in the development of the assignments. We very much appreciated Dr. Dean Brock, the chairman of the Department of Computer Science at UNC-Asheville who was in attendance at lectures and served as the faculty contact at UNC-Asheville. We also appreciate all the students in the Grid computing class of Spring 2007 who helped make the course a success. Class member Aurora Cain did the student portlet design shown in Figure 5.

## 7. REFERENCES

[1] Apon A., Mache J.,Yara Y., and Landrus K., Classroom exercises for grid services, *Proceedings of the Linux Cluster Institute International Conference on High Performance Computing*, Austin, Texas, May, 2004.

[2] Holliday M.A., Wilkinson B., House J., Daoud S., Ferner C., Geographically-distributed, assignment-structured undergraduate grid computing course, in *The 36th ACM Technical Symposium on Computer Science Education (SIGCSE2005)*, St. Louis, Missouri, pp. 206-210, February 23-27, 2005.

[3] Mache, J., and Apon, A., Grid computing in the undergraduate classroom topics, exercises and experiences, *Cluster Computing and the Grid 2005 (CCGrid)*, vol. 1, pp. 67- 73, May 2005.

[4] Mache, J., and Apon A., Teaching grid computing: topics, exercises, and experiences, *IEEE Transactions on Education*, Vol. 50, No. 1, pp. 3-9, Feb. 2007.

[5] Mache, J., Hands-on grid computing with Globus Toolkit 4, *Journal of Computing Sciences in Colleges*, Vol. 22, No. 2, pp. 99 – 100, Dec. 2006.

[6] National Science Foundation grant, “Collaborative: A Multi-Tier Model for Adaptation of Grid Technology to CS-based Undergraduate Curriculum,” ref. DUE # 0311473, PI: B. Ramamurthy, co-PI: B. Jayaraman, 2003-2006.

[7] National Science Foundation grant, “Collaborative Project: Adaptation of Globus Toolkit 3 Tutorials for Undergraduate Computer Science Students,” ref. DUE #0410966/#0411237, PIs: A. Apon and J. Mache, 2004-2007.

[8] National Science Foundation grant, “Introducing Grid Computing into the Undergraduate Curricula,” ref. DUE #0410667/0533334, PI: A. B. Wilkinson, co-PI M. Holliday, 2004-2007.

[9] The OGCE Portal Toolkit. http://www.collab-ogce.org/ogce2/

[10] PURSe: Portal-based User Registration Service. http://www.grids-center.org/solutions/purse/

[11] Wilkinson, B., and Ferner, C., Teaching grid computing in North Carolina: Two part series, *IEEE Distributed Systems Online*, vol. 7, no. 6&7, 2006, art. nos. 0606-o6003 and 0607-o7003.

[12] Wilkinson B., Holliday M., and Ferner C., Experiences in teaching a geographically distributed undergraduate grid computing course, *Proceedings of The Second International Workshop on Collaborative and Learning Applications of Grid Technology and Grid Education (Held in conjunction with CCGrid2005)*, May 9 - 12, 2005, Cardiff, United Kingdom.