The Grid, Virtual Organizations, and Problem-Solving Environments

David W. Walker
Department of Computer Science
Cardiff University
PO Box 916
Cardiff CF24 3XF
United Kingdom
david@cs.cf.ac.uk

1 Introduction
The Grid has been described as a distributed computing infrastructure for “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations”[1]. A virtual organization (VO) is a set of individuals and/or institutions with some common purpose or interest and that need to share their resources to further their objectives. These resources are typically computers, data, software, expertise, sensors, and instruments. A problem-solving environment (PSE) has been defined as “a computer system that provides all the computational facilities necessary to solve a target class of problems”[2]. For large-scale, complex, multi-disciplinary problems these facilities are usually distributed and owned by different collaborating organizations. Thus, for a significant class of problems a PSE can be viewed as the environment through which the end-users in a VO exploit Grid resources. In general, PSEs are accessed through a web interface and are thus very similar to “application portals.” PSEs can be used for collaborative computational science and engineering and tele-science applications. A good example is the Cactus open source PSE [3] that has been developed by an international collaboration of physicists and computational scientists to perform astrophysical simulations. The need for PSEs to support collaboration in distributed environments is widely recognized [4], however, these aspects introduce new types of complexity to the problem-solving process. In particular, end-users should not need to learn about generic middleware such as Globus to use a PSE. This talk will examine how “intelligence” in PSEs can deal with this complexity and help end-users in VOs make effective use of Grid resources.

2 Intelligent Infrastructure and User Support
In the current context “intelligence” refers to the ability of a piece of software to make reasonable, autonomous (or semi-autonomous) decisions on behalf of end-users, either individually or collectively. Examples of this type of intelligence include:

- Recommender systems that assist the user in determining the best way of solving their particular problems. Pythia-II combines knowledge discovery in databases (KDD) methodologies with recommender system technologies to assist users in
locating software to solve their problem, and in choosing from a number of alternative solution methods [5].

- Scheduling the components comprising a distributed application to achieve specified goals, within certain constraints. A scheduling goal for an individual end user might be to minimize execution time, whilst for the goal for a service provider (of compute cycles, storage, etc) might to maximize throughput. If the resources must be paid for then an end user might seek to minimize execution time subject to a cost constraint while a service provider might seek to maximize profit [6]. Achieving fair and efficient transparent access to resources is a difficult problem, for which the ability to monitor and predict the performance of both software and hardware is of key importance. The scheduler can be recast as a recommender system that presents a user with schedules that satisfy the user’s requirements and constraints, and assist the user in choosing one of them. This doesn’t provide complete transparency, and nor does it address the global scheduling problem.

- Intelligent interfaces seek to provide the end user with a view of the PSE that matches their degree of expertise and problem-solving requirements, as well as ensuring that access policies and security requirements are adhered to. Such an interface should also provide access to user support capabilities.

- Automatic componentization of legacy software, especially large libraries of numerical software. Validated legacy software represents a significant investment and it is important to be able to automatically “wrap” such software for use as components in distributed environments.

3 Closing Remarks
Abstraction and encapsulation are standard computer science techniques for shielding end-users of computing systems from complexities of the hardware and software. In collaborative and distributed systems additional complexities arise from pervasive heterogeneity and the dynamic nature of the distributed environment. The PSEs through which end-users interact with Grid resources must be intelligent enough to handle these complexities. The Semantic Web has been suggested as a way of allowing computers, users, and agent-based software systems to interact more effectively. The introduction of Semantic Web concepts into an intermediate layer between the Grid and PSEs may provide one mechanism upon which intelligent PSEs can be based. Currently agents play quite a limited role in scientific Grid applications. However, the agent abstraction may play a more significant role in the future of the Grid.