

# Brain Meets Brawn: Why Grid and Agents Need Each Other

Ian Foster  
Argonne National Laboratory  
& University of Chicago  
foster@mcs.anl.gov

Nicholas R. Jennings  
Electronics & Comp. Science  
University of Southampton  
nrj@ecs.soton.ac.uk

Carl Kesselman  
Information Sciences Institute  
University of Southern California  
carl@isi.edu

## Abstract

*The Grid and agent communities both develop concepts and mechanisms for open distributed systems, albeit from different perspectives. The Grid community has historically focused on “brawn”: infrastructure, tools, and applications for reliable and secure resource sharing within dynamic and geographically distributed virtual organizations. In contrast, the agents community has focused on “brain”: autonomous problem solvers that can act flexibly in uncertain and dynamic environments. Yet as the scale and ambition of both Grid and agent deployments increase, we see a convergence of interests, with agent systems requiring robust infrastructure and Grid systems requiring autonomous, flexible behaviors. Motivated by this convergence of interests, we review the current state of the art in both areas, review the challenges that concern the two communities, and propose research and technology development activities that can allow for mutually supportive efforts.*

## 1 Introduction

In open distributed systems, independent components cooperate to achieve individual and shared goals. Both individual components and the system as a whole are designed to cope with change and evolution in the number and nature of the participating entities. Such systems are important in many contexts, from large scientific collaborations to enterprise systems and sensor networks.

The Grid and agent communities are both pursuing the development of such open distributed systems, albeit from different perspectives. The Grid community [12] has historically focused on what we refer to here as “brawn”: interoperable infrastructure and tools for secure and reliable resource sharing within dynamic and geographically distributed virtual organizations (VOs) [14], and applications of the same to various resource federation scenarios. In contrast, those working on agents have focused on “brains,” i.e., on the development of concepts, methodologies, and algorithms for autonomous problem solvers that can act flexibly in uncertain and dynamic environments in order to achieve their aims and objectives [21]. A key component of this research is motivated by the fact that such agents are often required to form themselves into collectives (i.e., VOs) and act in a coordinated manner. This need to support aggregation

has, in turn, led to much research into rich and flexible mechanisms for managing such interactions.

As these two communities mature and turn their attention to fundamental problems of scope, both are encountering challenging problems in terms of scale and application. This maturation process is causing an increasing overlap in the problems that they address. Specifically, current Grid systems are somewhat rigid and inflexible in terms of their interoperation and their interactions, while agent systems are typically not engineered as serious distributed systems that need to scale, that are robust, and that are secure [34]. Nevertheless, each is working its way towards the others’ territory, as Grids seek to become more flexible and agile, and agent systems seek to be more reliable and scaleable.

Given this background, it is fruitful to examine work in these two domains, first to communicate to each community what has been done by the other, and second to identify opportunities for cross fertilization. We seek to take a first step towards that goal in this paper. To this end, we first review the state of the art in Grids and agents (Sections 2 and 3), compare and contrast the two approaches (Section 4), present a common vision of service-oriented architecture (Section 5), and conclude with a list of significant research challenges (Section 6).

Limited time and space require that we restrict ourselves in this article to the work being performed within the Grid and agents communities. Thus, we do not cover the highly relevant and interesting work pertaining to open distributed systems that can be found in other domains, including robotics, peer-to-peer networking, semantic web, distributed systems, artificial intelligence, and autonomic systems.

## 2 Grids

Grids aim to enable “resource sharing and coordinated problem solving in dynamic, multi-institutional VOs” [12]. In other words, Grids provide an infrastructure for federated resource sharing across trust domains. Much like the Internet on which they build, current Grids define protocols and middleware that can mediate access provided by this layer to discover, aggregate, and harness resources. These applications span a wide spectrum. Moreover, the standardization of the protocols and interfaces used to construct systems is an important part of the overall research and development program.

## 2.1 Technologies

Grid technologies have evolved through at least three distinct generations: early ad hoc solutions, de facto standards based on the Globus Toolkit (GT), and the current emergence of more formal Web services (WS)-based standards within the context of the Open Grid Services Architecture (OGSA) [13].

OGSA adopts WS standards such as Web Services Description Language (WSDL) as a basis for a service-oriented architecture within which arbitrary services can be defined, discovered, and invoked in terms of their interfaces rather than their implementations. This approach provides a basis for virtualization, interoperability, and composition.

The Grid community has participated in, and in some cases led, the development of WS specifications that address other Grid requirements. The WS-Resource Framework (WSRF) defines uniform mechanisms for defining, inspecting, and managing remote state, a crucial concern in many settings. WSRF mechanisms underlie work on service management (WSDM, in OASIS) and negotiation (WS-Agreement, in GGF), efforts that are crucial to the Grid vision of large-scale, reliable, and interoperable Grid applications and services. Other relevant efforts are aimed at standardizing interfaces to data, computers, and other classes of resources.

Work on Grid-related standards is driven by, and influences, the work of a vibrant open source community. GT (in its most recent instantiation, Web services-based and WSRF-compliant) provides basic middleware to create VOs, addressing such issues as specification and enforcement of VO wide policy, discovery, provisioning and management of services and resources, and federation, replication, discovery, and movement of data. At deployment, depending on available resources and planned applications, specific service implementations can be chosen and deployed, often in conjunction with other GT-based components.

Grid technology R&D has produced specifications and technologies for realizing service-oriented architectures according to robust distributed system principles. Global control mechanisms able to deal reliably with failure and adapt to changing environmental conditions and application concerns have been a lesser concern.

## 2.2 Applications

Early application drivers were largely from scientific computing [6, 10, 19], and included large-scale distributed computing [2, 15] (federation of computers), integration of large-scale data repositories (data grids [7]), collaboration [31], and tele-instrumentation [23, 26]. More recently, the technology has seen considerable uptake in industry as a means of addressing issues of virtualization and distributed system management [13].

GT is in production use across VOs integrating resources from 20-50 sites with thousands of computational and data resources, and is expected to scale to 100s of sites with 1000s of sites as a future goal. In the remainder of this section, we list a few examples to show the range and scope of Grid deployments.

The U.S. Network for Earthquake Engineering Simulation Grid (NEESgrid) connects experimental facilities (e.g., shake tables), data archives, computers, and a user community of earthquake engineers. Its service-oriented architecture defines standard interfaces for telepresence, monitoring, and control of remote scientific instruments, and for publishing, discovering, and accessing data produced by these instruments [26]. NEESgrid experiments have linked facilities at three sites and more than 50 remote participants.

Grid3 [15] links 28 sites with clusters totaling some 3000 processors. These resources are used by science communities from high energy physics, astronomy, biology, chemistry, and computer science for large-scale simulation and data analysis computations.

In contrast, Access Grid [31] is focused on interpersonal communication, via sharing of audio, video, and applications within collaborative spaces. Grid technologies are used in Access Grid for such purposes as security, discovery, and resource management.

Butterfly.net is creating a GT-based provisioning infrastructure for multiplayer online games, in which the demands for computation, storage, and network resources can vary dramatically as the popularity of games changes over time [24]. As a second example of a commercial Grid deployment, GlobeXplorer is using GT to support integration and processing of satellite image data [17].

Experiences with such applications reveal issues that must be addressed if Grids are to be scaled to larger communities, more diverse resources, and more complex applications. We review those challenges in Section 6.

## 3 Agent-Based Computing

An agent “is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its design objectives” [33]. In more detail [21], agents are: (i) clearly identifiable problem solving entities with well-defined boundaries and interfaces; (ii) situated (embedded) in a particular environment—they receive inputs related to the state of their environment through sensors and they act on the environment through effectors; (iii) designed to fulfill a specific role—they have particular objectives to achieve and have particular problem solving capabilities (services) that they can bring to bear to this end; (iv) autonomous—they have control both over their internal state and over their own behavior; and (v) capable of exhibiting flexible problem solving behavior in pursuit of their design objectives—they need

to be both reactive (able to respond in a timely fashion to changes that occur in their environment) and proactive (able to opportunistically adopt goals and take the initiative).

When adopting an agent-oriented view of the world, it soon becomes apparent that most problems require or involve multiple agents: to represent the decentralized nature of the problem, multiple loci of control, multiple perspectives, or competing interests. Moreover, these agents need to interact, either to achieve their individual objectives or to manage the dependencies that ensue from being situated in a common environment. Thus, in any given system there may be both cooperative and selfish agents whose aims are, respectively, to maximize the social welfare of the system and to maximize their own individual return. These interactions are built on some form of semantic integration (Section 2.3), may well involve trust relationships, and also include the traditional service discovery and invocation discussed above, as well as the more sophisticated social interactions related to the ability to cooperate, coordinate and negotiate about which services are performed by which agents at what time.

In the majority of cases, agents act to achieve objectives either on behalf of individuals (or companies) or as part of some wider problem solving initiative. (Note the similarity to the VO concept.) Thus, when agents interact there is typically some underpinning organizational context that defines the relationship among them. For example, agents may be peers working together in a team or one may be the manager of the other agents. To capture such links, agent systems typically have explicit constructs for modeling organizational relationships or roles such as peer, manager, or team member. In many cases, these relationships are subject to ongoing change: social interaction means existing relationships evolve (e.g., a team of peers may elect a leader) and new relations are created (e.g., a number of agents may form a VO to deliver a particular service that no one individual can offer). The temporal extent of these relationships can also vary enormously: from just long enough to deliver a particular service once, to a permanent bond.

Whatever the nature of the social process, there are two points that qualitatively differentiate agent interactions from those that occur in other computational models. First, agent-oriented interactions tend to be more sophisticated than in other contexts, dealing, for example, with notions of cooperation, coordination, and negotiation. Second, agents are flexible problem solvers, operating in an environment over which they have only partial control and observability. Thus, interactions need to be handled in a similarly flexible manner, and agents need the computational apparatus to make context-dependent decisions about the nature and scope of their interactions and to initiate (and respond to) interactions that were not foreseen at design time. The downside of

this autonomy and flexibility, however, is that it is difficult to ensure that desirable global behaviors emerge. To this end, a range of techniques (such as reinforcement learning, mechanism design, and electronic institutions) are often deployed to try and impose greater order.

Drawing these points together, Figure 1 shows that adopting an agent-oriented approach to system engineering means decomposing the problem into multiple, interacting, autonomous components that have particular objectives to achieve and are capable of performing particular services. The key abstraction models that define the agent-oriented mindset are agents, interactions and organizations. Finally, explicit structures and mechanisms are often used to describe and manage the complex and changing web of organizational relationships that exist between the agents.

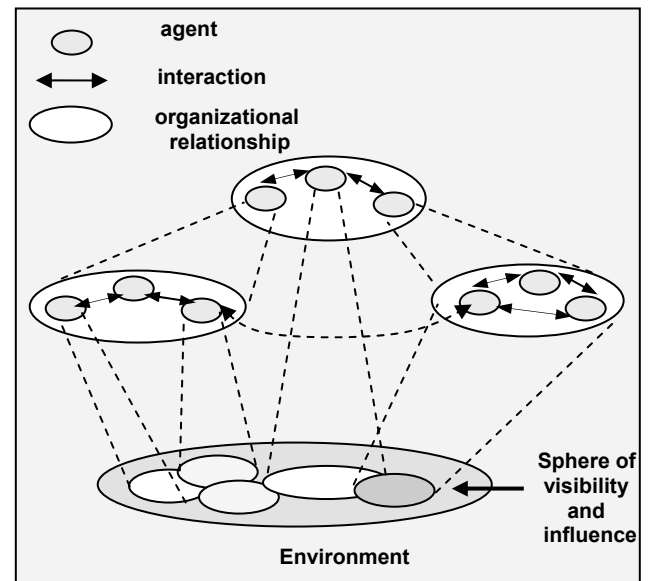


Figure 1: Canonical view of a multiagent system

### 3.1 Technologies

In contrast to Grid computing, there is less focus on identifiable agent technologies that can be used off the shelf to build applications. Traditionally, more attention has been given to theories and models of how agents can be developed and how they can communicate, cooperate, and negotiate. This work has resulted in the development of a range of algorithms that can be used both to build individual agents and to manage their interactions. In the former case, algorithms and architectures have been developed that enable an agent to plan an effective course of action to achieve a goal in uncertain and unpredictable environments, to adapt its behavior to its prevailing circumstances, and to strike an effective balance between being too responsive (and continually changing its aim such that no task is ever

completed) and too committed to its present course of action (such that more important activities are not dealt with in a timely fashion). In the latter case, algorithms have been developed that agents can use to achieve efficient negotiation outcomes, to form teams composed of the optimal set of parties, and to determine the degree of trust that should be placed in a particular agent, based upon its social and organizational relationships.

There has recently been an increasing trend towards making agent technology a serious basis for building complex, distributed systems. Several agent development environments support specific agent architectures and provide libraries of interaction protocols (e.g., JACK, JADE, Cougaar, and ZEUS), software engineering methodologies have been devised to analyze and design agent-based systems (e.g., Gaia, Tropos, and AUML), and there have been efforts to standardize various aspects of agent systems, such as inter-agent communication (e.g., FIPA, KQML). Moreover, as in the Grid community, there is an increasingly reliance on Web services and semantic web technologies for providing the computational infrastructure for such systems and an increasing acceptance of the importance of trust as a central issue in interaction.

### 3.2 Applications

Agent technology has been deployed in a number of isolated applications over the past ten years. However in the past few years the number and range of applications have increased significantly. In particular, many large companies are now interested in developing applications using agent technologies, and deployed applications exist for domains such as manufacturing, electronic commerce, process control, telecommunication systems, traffic and transportation management, information filtering and gathering, business process management, defense, entertainment and medical care [25].

## 4 Brains and Brawn

We see that a common thread underlies both agents and Grids, namely, the creation of communities or VOs bound together by a common goal or cause. Yet the two communities have focused on different aspects of this common problem. In the case of Grids, the primary concern has been the mechanisms by which communities form and operate. Thus, we see much effort devoted to how community standards are represented via explicit policy, how policy is enforced, how community members identify one another, how actions within the community are implemented, and how commitments by community members are specified, monitored and enforced. On the other hand, our understanding of how to use these mechanisms to create large-scale systems with stable collective behavior is less mature. For example, commonly used Grid tools provide uniform mechanisms

for accessing data on different storage systems, but not for the semantic integration of that data; for accessing service and resource state, but not for anticipating, detecting, and diagnosing problems implied by changes to that state; and for securely authenticating users and services, but not for inferring whether or not specific users or services can be trusted to perform specific actions. To this extent, Grids are all brawn and no brain.

Agents also focus on creating community. Out of the flexible local decision making of system components, sensible community wide behaviors emerge through rich social interactions and explicit organizational structures. However in building all this flexibility and sophistication, scant attention has been paid to how these tasks should be performed in realistic distributed environments. For example, agent frameworks provide sophisticated internal reasoning capabilities, but offer no support for secure interaction or service discovery; cooperation algorithms produce socially optimal outcomes, but assume the agents have complete knowledge of all outcomes that any potential grouping can produce; and negotiation algorithms achieve optimal outcomes for the participating agents, but assume that all parties in the system are known at the outset of the negotiation and will not change during the system's operation. Thus, one may say that agents are all brain and no brawn.

Clearly, neither situation is ideal: for Grids to be effective in their goals, they must be imbued with flexible, decentralized decision making capabilities. Likewise, agents need a robust distributed computing platform that allows them to discover, acquire, federate, and manage the capabilities necessary to execute their decisions. In other words, there are good opportunities for exploiting synergies between Grid and agents.

One approach to exploiting such synergies might be a simple layering of the technologies, i.e., to implement agent systems on top of Grid mechanisms. However, it seems more likely that the true benefits of an integrated Grid/agent approach will only be achieved via a more fine-grain intertwining of the two technologies, with Grid technologies becoming more agent-like and agent-based systems becoming more Grid-like.

As an early example of such a tighter coupling, we point to work on agent-based resource selection, in which re-enforcement-based learning is used to drive the assignment of tasks to resources [16]. In this case, the "agent" (i.e., the logic used to make the task assignment decisions) uses Grid functions for status monitoring, resource discovery, and task submission. The agent, in turn, provides a valuable Grid function, with the collection of agents implementing a robust global resource management behavior that might not otherwise be achieved. A second example is the use of automated negotiation techniques (specifically, various forms of auctions) to allocate resources in Grid systems [32]. Here, designers evaluate the effectiveness of both commodity

market and Vickery auction protocols to the problem of allocating resources within a distributed system. This example also shows how techniques familiar to agents researchers can be integrated with other more standard components within a Grid architecture.

This level of integration will undoubtedly create new challenges for both agents and Grids. However, the result could be frameworks for constructing robust, large-scale, agile distributed systems that are qualitatively and quantitatively superior to the best current practice today.

## 5 Robust Agile Service-Oriented Systems

Having described key agent and Grid concepts, we now draw the two parallel lines of research together to highlight their commonalities and complementarities.

### 5.1 Autonomous Services

A core unifying concept that underlies Grids and agent systems is that of a *service*: an entity that provides a capability to a *client* via a well-defined message exchange [4]. Within third-generation Grids, service interactions are structured via Web service mechanisms, and thus all entities are services. However, while every agent can be considered a service (in that it interacts with other agents and its environment via message exchanges), we might reasonably state that not every Grid service is necessarily an agent (in that it may not participate in message exchanges that exhibit flexible autonomous actions).

This notion of *autonomous action* is thus central to the question of how agents and Grids can interoperate. To illustrate the issues, let us consider a service that encapsulates a database. In a local area network, we might find a version of this service that responds to requests to “read a record” or “write a record.” Such an implementation does not exhibit autonomous behavior.

On the other hand, in a more distributed, administratively heterogeneous, and failure-prone environment, the implementation of such a service might exhibit more sophisticated behavior. For example, the database might be replicated, with the number of replicas determined dynamically by knowledge-based models of system reliability and performance. Distributed negotiation protocols might be used to establish the query throughput achievable on individual copies, such that community throughput is optimized. Finally, distributed planning and scheduling algorithms might be used to map queries to specific database replicas so as to minimize the latency of user requests. In all these cases, a robust database service, designed to operate in an open distributed system, is exhibiting flexible autonomous actions (in the sense that its behaviors are not driven solely by a client request, but also by other considerations, including local policies and the outcomes of negotiations with the client). In short, such services will exhibit agent behavior.

### 5.2 Rich Service Models

Both agent and Grid systems consist of *dynamic* and *stateful* services. The underlying service model is dynamic in that new services can be created and destroyed over the lifetime of the system. Here an important contribution of Grid technologies is a robust lifetime and naming model for dynamic services [13]. Implicit in this model are the notion of service failure and the definition of a scalable distributed systems semantics. In contrast, agent-based systems rarely consider such issues, but they could clearly benefit from exploiting this approach to representing and managing dynamic services.

Statefulness is another important aspect of the service model. A stateful service (or, more-or-less equivalently, a *resource* [11]) has internal state that persists over multiple interactions. It can often be useful to make this state externally visible, so that, for example, another participant in a distributed system can determine the current load on a server, the policies that govern access to a service, and/or the schema(s) supported by a database. Again, Grid technologies have addressed this issue, defining a general model for representing and querying service state [11]. This model includes mechanisms for describing state “lifetime”, as well as a means of specifying and enforcing policy with respect to access and modification.

The Grid state model defines how state is represented and accessed, but does not speak to the structure or semantics of the state that is thus exposed. Typical practice is to define state in terms of fixed schema or attributes. In contrast, agent systems address semantics but do not provide a consistent state model. An integrated approach can allow for the publication of richer semantic information within the Grid state model, thus enhancing the ability of applications to discover, configure, and manage services in an interoperable manner [18].

### 5.3 Negotiation and Service Contracts

Negotiation is emblematic of the brain/brawn schism between current Grid and agent systems. In general, it cannot be assumed that a service will actually provide a particular capability to a user: a provider may be unable or unwilling to provide the service to a putative consumer. Hence, if the system is to have any type of predictable behavior, it becomes necessary to obtain commitments (contracts) about the willingness to provide a service and the characteristics, or quality, of its provision.

Given the ability to provision a resource to provide a desired level of service, we are faced with the question of exactly what levels of service can and should be obtained. The process by which this is determined will necessarily be some form of negotiation, since the autonomous entities involved need to come to a mutually acceptable agreement on the matter. If this negotiation is successful (i.e., both parties come to an agreement) then the outcome

of the procurement is a contract (service level agreement) between the service provider and the service consumer.

This negotiation can be arranged in many different ways; there are millions of protocols, with varying properties, and agent researchers have invested significant effort in determining which protocols are appropriate in which circumstances [9]. In this context, the negotiation is driven by the operational policy of both the service provider and the service consumer. Specifically, policy terms to be considered may involve aspects such as the current load, the identity and reputation of the requestor, and the requestor's ability to pay.

The use of negotiation as a means of establishing service contracts is a topic of considerable interest in both the agent [22] and Grid [8] communities. One promising approach within Grids has been to represent agreement as the creation of a shared policy statement and to define robust extensible protocols for exchanging and agreeing to policy terms. Creating these agreements in the face of a Byzantine failure model can be complex. Having designed such protocols, the next step is to determine the strategy that the system components should adopt to achieve their policy objectives. Strategies can vary from the simple (e.g., an agent bidding its true valuation for a service) to the complex (reasoning about the other participants and their likely strategies).

#### 5.4 Virtual Organization Management

A common interaction modality in both Grid and agent systems occurs when several agents come together to form a new VO. Such VOs can be viewed as a form of dynamic service composition: a number of initially distinct entities come together, under a set of operating conditions, to form a new entity that offers a new service. In such cases, one of the key challenges is for the participating agents to determine who else should be involved in the coalition and what their various roles and responsibilities should be. Again, this activity typically involves negotiation among participants, in this case to determine a mutually acceptable agreement concerning the division of labor and responsibilities.

Dynamic creation also raises the issue of service discovery. Experience in the Grid community indicates that this discovery should not simply be on the basis of service type, but rather should incorporate notions of service state and should be based on an understanding of the capabilities of the service (i.e., semantics). While Grid technologies provide the means for describing and grouping services, these higher level matchmaking and discovery capabilities are not currently part of Grid infrastructure. Fortunately, this is an area where much work has been done in the space of agents, and thus incorporation of this technology would do much to improve matters. This integration may have an impact on how state is represented and how services are organized.

#### 5.5 Authentication, Trust, and Policy

As discussed in Section 5.2, the association of identity with dynamically created services has long been an integral part of Grid infrastructure. A common approach to this problem is to map identities into a global namespace and then apply delegation as a means for building federated namespaces for dynamically created entities. More recent work has focused on the application of richer policy statements and the creation of community based authorization and assertion authorities [27].

Also fundamental to the creation of collaboration and community, and building upon the aforementioned notions of authentication, are notions of trust. The effective management of trust and policy within a community, like VO formation, requires flexible, autonomous mechanisms able to consider, when organizing communities, not only the semantics of policy statements but also the ability to negotiate policy terms and to manage restricted delegation of rights.

As with other aspects of agents and Grids, we expect to see the adaptation of agent algorithms and technologies as they incorporate policy specification and enforcement into their basic operations and we expect to see Grid algorithms make use of some of the richness of the various agent trust and reputation models that have been developed [28]. We also expect that the types of policy statements made, along with how they are disseminated and applied, will evolve as agent-based techniques become more completely integrated into Grids. For example, reputation-based authentication mechanisms, which lend themselves to agent-based implementations, show great promise in the Grid environment.

### 6 Ten Research Problems

We conclude by outlining ten areas (in no particular order) in which research is needed to realize an integrated agent-Grid approach to open distributed systems.

**Service architecture.** The convergence of agent and Grid concepts and technologies will be accelerated if we can define an integrated service architecture providing a robust foundation for autonomous behaviors. This architecture would define baseline interfaces and behaviors supporting dynamic and stateful services, and a suite of higher-level interfaces and services codifying important negotiation, monitoring, and management patterns. The definition of an appropriate set of such architectural elements is an important research goal in its own right, and, in addition, can facilitate the creation, reuse, and composition of interoperable components.

**Trust negotiation and management.** All but the most trivial distributed systems involve interactions with entities (services) with whom one does not have perfect trust. Thus, authorization decisions must often be made in

the absence of strong existing trust relationships. Grid middleware addresses secure authentication, but not the far harder problems of establishing, monitoring, and managing trust in a dynamic, open, multi-valent system. We need new techniques for expressing and reasoning about trust. Reputation mechanisms [29] and the ability to integrate assertions from multiple authorities (“A says M can do X, but B disagrees”) will be important in many contexts, with the identity and/or prior actions of an entity requesting some action or asserting some fact being as important as other metrics, such as location or willingness to pay. Trust issues can also impinge on data integration, in that our confidence in the “data” provided by an entity may depend on our trust in that entity, so that, for example, our confidence in an assertion “A says M is green” depends on our past experiences with A.

**System management and troubleshooting.** Grid technologies make it feasible to access large numbers of resources securely, reliably, and uniformly. However, the coordinated management of these resources requires new abstractions, mechanisms, and standards for the quasi-automated (“autonomic” [20]) management of the ensemble—despite multiple, perhaps competing, objectives from different parties, and complex failure scenarios. A closely related problem is troubleshooting, i.e., detecting, diagnosing, and ultimately responding to the unexpected behavior of an individual component in a distributed system, or indeed of the system as a whole. This requirement will motivate the development of robust and secure logging and auditing mechanisms. The registration, discovery, monitoring, and management of available logging points, and the development of techniques for detecting and responding to “trouble” (e.g., overload or fraud), remain open problems. We also require advances in the summarization and explanation (e.g., visualization) of large-scale distributed systems.

**Negotiation.** We have already discussed negotiation at some length; here we simply note that major open problems remain in this vital area.

**Service composition.** The realization of a specific user or VO requirement may require the dynamic composition of multiple services. Web service technologies define conventions for describing service interfaces and workflows, and WSRF provides mechanisms for inspecting service state and organizing service collections. Yet we need far more powerful techniques for describing, discovering, composing, monitoring, managing, and adapting such service collections.

**VO formation and management.** While the notion of a VO seems to be intuitive and natural, we still do not have clear definitions of what constitutes a VO or well-defined procedures for deciding when a new VO should be formed, who should be in that VO, what they should do,

when the VO should be changed, and when the VO should ultimately be disbanded.

**System predictability.** While open distributed systems are inherently unpredictable, it can be important to provide guarantees about system performance (e.g., liveness or safety properties, or stochastic performance boundaries). However, such guarantees require a deeper understanding of emergent behavior in complex systems.

**Human-computer collaboration.** Many VOs will be hybrids in which some problem solving is undertaken by humans and some by programs. These components must interwork in a seamless fashion to achieve their aims. New collaboration models are necessary to capture the rich social interplay in such hybrid teams.

**Evaluation.** Meaningful comparison of new approaches and technologies requires the definition of appropriate benchmarks and challenge problems and the creation of environments in which realistic evaluation can occur. Perhaps the single most effective means of advancing agent-Grid integration might be the definition of appropriately attractive challenge problems. Such problems should demand both the brawn of Grid and the brains of agents, and define rigorous metrics that can be used to drive the development in both areas. Potential challenge problems might include the distributed monitoring and management of large-scale Grids, and robust and long-lived operation of agent applications.

Evaluation can occur in both simulated and physical environments. Rapid progress has been made in simulation systems for both agents and Grids (e.g., [30]). Production deployments such as Grid3 [15], TeraGrid [5], and NEESgrid [26], and testbeds such as PlanetLab [1], are potentially available as experimental platforms for the evaluation of converged systems, for example within the context of the challenge problems just mentioned.

**Semantic integration.** Open distributed systems involve multiple stakeholders that interact to procure and deliver services. Meaningful interactions are difficult to achieve in any open system because different entities typically have distinct information models. Advances are required in such interrelated areas as ontology definition, schema mediation, and semantic mediation [3]. Again, issues of trust and cost have vital roles to play.

## Acknowledgments

The work of the first author was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38. The second author acknowledges the support of the EPSRC project “Virtual organisations for e-Science” (GR/S62710/01).

## References

1. Bavier, A., Bowman, M., Chun, B., Culler, D., Karlin, S., Muir, S., Peterson, L., Roscoe, T., Spalink, T. and Wawrzoniak, M., Operating System Support for Planetary-Scale Network Services. *Networking Systems Design and Implementation (NSDI)*, 2004.
2. Beiriger, J., Johnson, W., Bivens, H., Humphreys, S. and Rhea, R., Constructing the ASCI Grid. *9th IEEE International Symposium on High Performance Distributed Computing*, 2000, IEEE Computer Society Press.
3. Berners-Lee, T., Hendler, J. and Lassila, O. The Semantic Web. *Scientific American*, 284 (5). 34-43. 2001.
4. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. Web Services Architecture. W3C, Working Draft <http://www.w3.org/TR/2003/WD-ws-arch-20030808/>, 2003.
5. Catlett, C. The TeraGrid: A Primer, 2002. [www.teragrid.org](http://www.teragrid.org).
6. Catlett, C. and Smarr, L. Metacomputing. *Communications of the ACM*, 35 (6). 44-52. 1992.
7. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. *J. Network and Computer Applications*, 23 (3). 187-200. 2000.
8. Czajkowski, K., Foster, I. and Kesselman, C. Resource and Service Management. *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, 2004.
9. Dash, R.K., Parkes, D.C. and Jennings, N.R. Computational Mechanism Design: A Call to Arms. *IEEE Intelligent Systems*, 18 (6). 40-47. 2003.
10. DeFanti, T., Foster, I., Papka, M., Stevens, R. and Kuhfuss, T. Overview of the I-WAY: Wide Area Visual Supercomputing. *International Journal of Supercomputer Applications*, 10 (2). 123-130. 1996.
11. Foster, I., Frey, J., Graham, S., Tuecke, S., Czajkowski, K., Ferguson, D., Leymann, F., Nally, M., Storey, T. and Weerawaranna, S. Modeling Stateful Resources with Web Services. Globus Alliance, 2004.
12. Foster, I. and Kesselman, C. (eds.). *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*. Morgan Kaufmann, 2004.
13. Foster, I., Kesselman, C., Nick, J.M. and Tuecke, S. Grid Services for Distributed Systems Integration. *IEEE Computer*, 35 (6). 37-46. 2002.
14. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 15 (3). 200-222. 2001.
15. Foster, I. and others, The Grid2003 Production Grid: Principles and Practice. *IEEE International Symposium on High Performance Distributed Computing*, 2004, IEEE Computer Science Press.
16. Galstyan, A., Czajkowski, K. and Lerman, K., Resource Allocation in the Grid Using Reinforcement Learning. *International Conference on Autonomous Agents and Multiagent Systems*, 2004.
17. Gentzsch, W. Enterprise Resource Management: Applications in Research and Industry. *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, Morgan Kaufmann, 2004.
18. Goble, C.A., De Roure, D., Shadbolt, N.R. and Fernandes, A. Enhancing Services and Applications with Knowledge and Semantics. *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, Morgan-Kaufmann, 2004.
19. Grimshaw, A., Weissman, J., West, E. and E. Lyot, J. Metasystems: An Approach Combining Parallel Processing and Heterogeneous Distributed Computing Systems. *Journal of Parallel and Distributed Computing*, 21 (3). 257-270. 1994.
20. Horn, P. The IBM Vision for Autonomic Computing. IBM, 2001. [www.research.ibm.com/autonomic/manifesto](http://www.research.ibm.com/autonomic/manifesto).
21. Jennings, N.R. An agent-based approach for building complex software systems. *Communications of the ACM*, 44 (4). 35-41. 2001.
22. Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Sierra, C. and Wooldridge, M. Automated negotiation: prospects, methods and challenges. *Intl. J. of Group Decision and Negotiation*, 10 (2). 199-215. 2001.
23. Johnston, W. Realtime Widely Distributed Instrumentation Systems. Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 75-103.
24. Levine, D. and Wirt, M. Interactivity with Scalability: Infrastructure for Multiplayer Games. *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, Morgan Kaufmann, 2004.
25. Luck, M., McBurney, P. and Preist, C. Agent technology: Enabling Next Generation Computing. *AgentLink*. 2003.
26. Pearlman, L., Kesselman, C., Gullapalli, S., Spencer, B.F., Futrelle, J., Ricker, K., Foster, I., Hubbard, P. and Severance, C., Distributed Hybrid Earthquake Engineering Experiments: Experiences with a Ground-Shaking Grid Application. *13th IEEE International Symposium on High Performance Distributed Computing*, 2004, NEESgrid.
27. Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., A Community Authorization Service for Group Collaboration. *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.
28. Ramchurn, S.D., Huynh, D. and Jennings, N.R. Trust in Multiagent Systems. *The Knowledge Engineering Review*. 2004.
29. Resnick, P., Zeckhauser, R., Friedman, E. and Kuwabara, K. Reputation Systems. *Communications of the ACM*, 43 (12). 45-48. 2000.
30. Song, H., Liu, X., Jakobsen, D., Bhagwan, R., Zhang, X., K., Taura and Chien, A., The MicroGrid: A Scientific Tool for Modeling Computational Grids. *SC 2000*, 2000, IEEE Computer Society Press.
31. Stevens, R. Group-Oriented Collaboration: The Access Grid Collaboration System. *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, Morgan Kaufmann, 2004.
32. Wolski, R., Brevik, J., Plank, J. and Bryan, T. Grid Resource Allocation and Control Using Computational Economics. Berman, F., Fox, G. and Hey, T. eds. *Grid Computing: Making the Global Infrastructure a Reality*, Wiley and Sons, 2003, 747-772.
33. Wooldridge, M. Agent-based software engineering. *IEE Proc. Software Engineering*, 144. 26-37. 1997.
34. Wooldridge, M. and Jennings, N.R. Software Engineering with Agents: Pitfalls and Pratfalls. *IEEE Internet Computing*, 3 (3). 20-27. 1999.