Research Agenda for the Semantic Grid:

A Future e-Science Infrastructure

December 2001

David De Roure
Nicholas Jennings
Nigel Shadbolt

Report commissioned for EPSRC/DTI Core e-Science Programme

Status of this document

This report was commissioned for EPSRC/DTI Core e-Science Programme by Professor A. G. Hey in June 2001. A working version of this document was released for comment with limited circulation to the UK Research Councils e-Science Programme Grid Technical Advisory Group in July 2001. The current document supersedes this earlier version.

The authors are based in the Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK and may be contacted via David De Roure on dder@ecs.soton.ac.uk

Acknowledgements

We are grateful to Mark Baker (co-chair of the IEEE Task Force in Cluster Computing) for his contributions to Section 3. Thanks also to Vijay Dialani for his input, and to the UK's Grid Technical Advisory Group members and others who provided comments on the first draft of this report, and to Luc Moreau, Mark Greenwood, Nick Gibbins and Rob Allan for commenting on the second draft. We are grateful to Epistemics Ltd for management of the contract.

Research Agenda for the Semantic Grid: A Future e-Science Infrastructure

David De Roure, Nicholas Jennings and Nigel Shadbolt

Executive Summary

e-Science offers a promising vision of how computer and communication technology can support and enhance the scientific process. It does this by enabling scientists to generate, analyse, share and discuss their insights, experiments and results in a more effective manner. The underlying computer infrastructure that provides these facilities is commonly referred to as the Grid. At this time, there are a number of grid applications being developed and there is a whole raft of computer technologies that provide fragments of the necessary functionality. However there is currently a major gap between these endeavours and the vision of e-Science in which there is a high degree of easy-to-use and seamless automation and in which there are flexible collaborations and computations on a global scale. To bridge this practice–aspiration divide, this report presents a research agenda whose aim is to move from the current state of the art in e-Science infrastructure, to the future infrastructure that is needed to support the full richness of the e-Science vision. Here the future e-Science research infrastructure is termed the Semantic Grid (Semantic Grid to Grid is meant to connote a similar relationship to the one that exists between the Semantic Web and the Web).

In more detail, this document analyses the state of the art and the research challenges that are involved in developing the computing infrastructure needed for e-Science. In so doing, a conceptual architecture for the Semantic Grid is presented. This architecture adopts a service-oriented perspective in which distinct stakeholders in the scientific process provide services to one another in various forms of marketplace. The view presented in the report is holistic, considering the requirements of e-Science and the e-Scientist at the data/computation, information and knowledge layers. The data, computation and information aspects are discussed from a distributed systems viewpoint and in the particular context of the Web as an established large scale infrastructure. A clear characterisation of the knowledge grid is also presented. This characterisation builds on the emerging metadata infrastructure with knowledge engineering techniques. These techniques are shown to be the key to working with heterogeneous information and also to working with experts and establishing communities of e-Scientists. The underlying fabric of the Grid, including the physical layer and associated technologies, is outside the scope of this document.

Having completed the analysis, the report then makes a number of recommendations that aim to ensure the full potential of e-Science is realised and that the maximum value is obtained from the endeavours associated with developing the Semantic Grid. These recommendations relate to the following aspects:

- The research issues associated with the technical and conceptual infrastructure of the Semantic Grid;
- The research issues associated with the content infrastructure of the Semantic Grid;
- The bootstrapping activities that are necessary to ensure the UK's grid and e-Science infrastructure is widely disseminated and exemplified;
- The human resource issues that need to be considered in order to make a success of the UK e-Science and grid efforts;
- The issues associated with the intrinsic process of undertaking e-Science;
- The future strategic activities that need to be undertaken to maximise the value from the various Semantic Grid endeavours.

Document revision history

Draft 0.1	25-5-2001	Section 2
Draft 0.2	2-6-2001	Sections 1 and 2, remaining sections in outline
Draft 0.3	7-6-2001	Sections 1, 2 and 5, remaining sections in outline
Draft 0.4	18/28-6-2001	Section 5 extended
Draft 0.5	29-6-2001	Integrate various sections
Draft 0.6	2-7-2001	Included draft of section 4
Draft 0.7	7-7-2001	Included draft of section 3
Draft 0.8	9-7-2001	Reworked references, topped and tailed sec 3 & 4
Draft 0.9	10-7-2001	Refined sections 3, 4 and 5
	11-7-2001	Distributed to TAG
Draft 1.0	10-9-2001	Added section 2.3 and refined section 2
Draft 1.1	17-9-2001	Revised scenario and updated section 3. Changed 5.4 in line with 2.3 and added 5.5 – included TAG comments relevant to section 5
Draft 1.2	20-10-2001	Consolidated 1.1 and updated section 4
Draft 1.3	9-11-2001	Minor revisions throughout, restructuring in section 4
Draft 1.4	19-11-2001	Minor revisions throughout
Draft 1.5	05-12-2001	Added Recommendations
Draft 1.6	06-12-2001	Minor revisions throughout based on comments
Draft 1.7	10-12-2001	Minor revisions throughout
Draft 1.8	12-12-2001	Release to Architecture Task Force
Draft 1.9	21-12-2001	Release to TAG

Contents

1. Introduction

1.1 Motivation

Scientific research and development has always involved large numbers of people, with different types and levels of expertise, working in a variety of roles, both separately and together, making use of and extending the body of knowledge. In recent years, however, there have been a number of important changes in the nature and the process of research. In particular, there is an increased emphasis on collaboration between large teams, an increased use of advanced information processing techniques, and an increased need to share results and observations between participants who are widely dispersed. When taken together, these trends mean that researchers are increasingly relying on computer and communication technologies as an intrinsic part of their everyday research activity. At present, the key communication technologies are predominantly email and the Web. Together these have shown a glimpse of what is possible; however to more fully support the e-Scientist the next generation of technology will need to be much richer, more flexible and much easier to use. Against this background, this report focuses on the requirements, the design and implementation issues, and the research challenges associated with developing a computing infrastructure to support e-Science.

The computing infrastructure for e-Science is commonly referred to as *the Grid* [Foster98] and this is, therefore, the term we will use here. This terminology is chosen to connote the idea of a 'power grid': namely that e-Scientists can plug into the e-Science computing infrastructure like plugging into a power grid. An important point to note however is that the term 'grid' is sometimes used synonymously with a networked, high performance computing infrastructure. While this aspect is certainly an important and exciting enabling technology for future e-Science, it is only a part of a much larger picture that also includes information handling and support for knowledge within the e-scientific process. It is this broader view of the e-Science infrastructure that we adopt in this document and we refer to this as the *Semantic Grid*. Our view is that as the Grid is to the Web, so the Semantic Grid is to the Semantic Web. Thus the Semantic Grid is characterised by an open system, with a high degree of automation, that supports flexible collaboration and computation on a global scale.

The grid metaphor intuitively gives rise to the view of the e-Science infrastructure as a set of services that are provided by particular individuals or institutions for consumption by others. Given this, and coupled with the fact that many research and standards activities are embracing a similar view [WebServices01], we adopt a *service-oriented view* of the Grid throughout this document (see section 2 for a more detailed justification of this choice). This view is based upon the notion of various *entities* providing *services* to one another under various forms of *contract* (or service level agreement).

Given the above view of the scope of e-Science, which includes information and knowledge, it has become popular to conceptualise the computing infrastructure as consisting of three conceptual layers[1]:

- **Data/computation**
  This layer deals with the way that computational resources are allocated, scheduled and executed and the way in which data is shipped between the various processing resources. It is characterised as being able to deal with large volumes of data, providing fast networks and presenting diverse resources as a single metacomputer (i.e. a single virtual computer). In terms of technology, this layer shares much with the body of research that has been undertaken into distributed computing systems. In the context of this document, we will discuss a variety of frameworks that are, or could be, deployed in grid computing at this level. The data/computation layer builds on the physical 'grid fabric', i.e. the underlying network and computer infrastructure, which may also interconnect scientific equipment. Here data is understood as uninterpreted bits and bytes.

- **Information**
  This layer deals with the way that information is represented, stored, accessed, shared and maintained. Given its key role in many scientific endeavours, the World Wide Web (WWW) is the obvious point of departure for this level. Thus in the context of this document, we will consider the extent to which the Web meets the e-Scientists' information requirements. In particular, we will pay attention to the current developments in Web research and standards and identify gaps where the needs of e-Scientists are not being met. Here information is understood as data equipped with meaning.

- **Knowledge**
  This layer is concerned with the way that knowledge is acquired, used, retrieved, published and maintained to assist e-Scientists to achieve their particular goals and objectives. In the context of this document, we review the state of the art in knowledge technologies for the Grid, and identify the major research issues that still need to be addressed. Here knowledge is understood as information applied to achieve a goal, solve a problem or enact a decision.

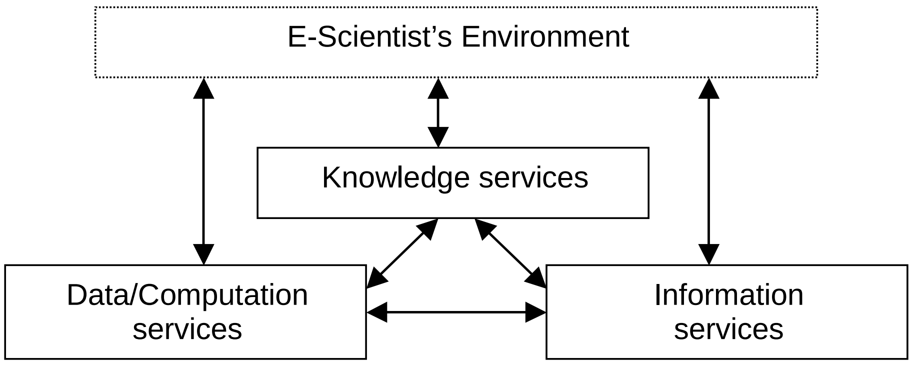

Figure 1.1: Three layered architecture viewed as services

[1] The three layer grid vision is attributed to Keith G. Jeffery of CLRC, who introduced it in a paper for the UK Research Councils Strategic Review in 1999.

We believe this structuring is compelling and we have, therefore, adopted it to structure this report. We also intend that this approach will provide a reader who is familiar with one layer with an introduction to the others, since the layers also tend to reflect distinct research communities and there is a significant need to bridge communities. However, there are a number of observations and remarks that need to be made. Firstly, all grids that have or will be built have some element of all three layers in them. The degree to which the various layers are important and utilised in a given application will be domain dependent – thus in some cases, the processing of huge volumes of data will be the dominant concern, while in others the knowledge services that are available will be the overriding issue. Secondly, this layering is a conceptual view on the system that is useful in the analysis and design phases of development. However, the strict layering may not be carried forward to the implementation for reasons of efficiency. Thirdly, the service-oriented view applies at all the layers. Thus there are services, producers, consumers and contracts at the computational layer, at the information layer and at the knowledge layer (figure 1.1).

Fourthly, a (power) grid is useless without appliances to plug in. Confining the infrastructure discussion to remote services runs the risk of neglecting the interface, i.e. the computers, devices and apparatus with which the e-Scientist interacts. While virtual supercomputers certainly offer potential for scientific breakthrough, trends in computer supported cooperative work (CSCW), such as embedded devices and collaborative virtual environments, also have tremendous potential in facilitating the scientific process. Whereas the grid computing literature has picked up on visualisation [Foster99] and virtual reality (VR) [Leigh99], comparatively little attention has been paid to pervasive computing and augmented reality – what might be called the 'smart laboratory'. This document addresses this omission.

With this context established, the next sub-section introduces a grid application scenario that we will use to motivate and explain the various tools and techniques that are available at the different grid levels.

1.2 Motivating Scenario

At this time, the precise set of requirements on the e-Science infrastructure are not clear since comparatively few applications have been envisaged in detail (let alone actually developed). While this is certainly going to change over the coming years, it means the best way of grounding the subsequent discussion on models, tools and techniques is in terms of a scenario. To this end, we will use the following scenario to motivate the discussion in this document.

This scenario is derived from talking with e-Scientists across several domains including physical sciences. It is not intended to be domain-specific (since this would be too narrow) and at the same time it cannot be completely generic (since this would not be detailed enough to serve as a basis for grounding our discussion). Thus it falls somewhere in between. Nor is the scenario science fiction – these practices exist today, but on a restricted scale and with a limited degree of automation. The scenario itself (figure 1.2) fits with the description of grid applications as "coordinated resource sharing and problem solving among dynamic collections of individuals" [Foster01].

Scenario

The sample arrives for analysis with an ID number. The technician logs it into the database and the information about the sample appears (it had been entered remotely when the sample was taken). The appropriate settings are confirmed and the sample is placed with the others going to the analyser (a piece of laboratory equipment). The analyser runs automatically and the output of the analysis is stored together with a record of the parameters and laboratory conditions at the time of analysis.

The analysis is automatically brought to the attention of the company scientist who routinely inspects analysis results such as these. The scientist reviews the results from their remote office and decides the sample needs further investigation. They request a booking to use the High Resolution Analyser and the system presents configurations for previous runs on similar samples; the scientist selects the appropriate parameters. Prior to the booking, the sample is taken to the analyser and the equipment recognizes the sample identification. The sample is placed in the equipment which configures appropriately, the door is locked and the experiment is monitored by the technician by live video then left to run overnight; the video is also recorded, along with live data from the equipment. The scientist is sent a URL to the results.

Later the scientist looks at the results and, intrigued, decides to replay the analyser run, navigating the video and associated data. They then press the “query” button and the system summarises previous related analyses reported internally and externally, and recommends other scientists who have published work in this area. The scientist finds that their results appear to be unique.

The scientist requests an agenda item at the next research videoconference and publishes the experimental data for access by their colleagues (only) in preparation for the meeting. The meeting decides to make the analysis available for the wider community to look at, so the scientist then logs the analysis and associated metadata into an international database and provides some covering information. Its provenance is recorded. The availability of the new data prompts other automatic processing and a number of databases are updated; some processing of this new data occurs.

Various scientists who had expressed interest in samples or analyses fitting this description are notified automatically. One of them decides to run a simulation to see if they can model the sample, using remote resources and visualizing the result locally. The simulation involves the use of a problem solving environment (PSE) within which to assemble a range of components to explore the issues and questions that arise for the scientist. The parameters and results of the simulations are made available via the public database. Another scientist adds annotation to the published data.

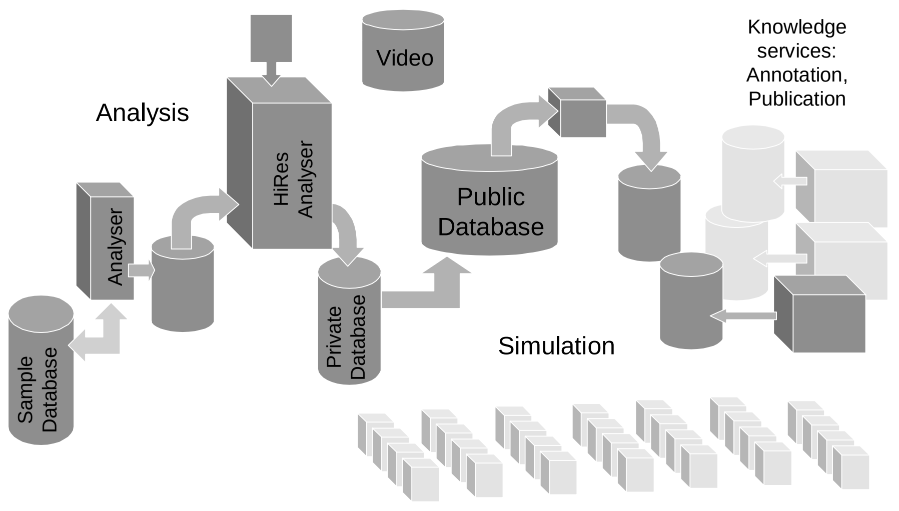

Figure 1.2: Workflow in the scenario

This scenario draws out a number of underlying assumptions and raises a number of requirements that we believe are broadly applicable to a range of e-Science applications:

- Storage. It is important that the system is able to store and process huge volumes of content in a timely and efficient fashion.
- Ownership. Different stakeholders need to be able to retain ownership of their own content and processing capabilities, but there is also a need to allow others access under the appropriate terms and conditions.
- Provenance. Sufficient information is stored so that it is possible to repeat the experiment, re-use the results, or provide evidence that this data was produced at this time (the latter may involve a third party).
- Transparency. Users need to be able to discover, transparently access and process relevant content wherever it may be located in the Grid.
- Communities. Users should be able to form, maintain and disband communities of practice with restricted membership criteria and rules of operation.
- Fusion. Content needs to be able to be combined from multiple sources in unpredictable ways according to the users' needs; descriptions of the sources and content will be used to combine content meaningfully.
- Conferencing. Sometimes it is useful to see the other members of the conference, and sometimes it is useful to see the artefacts and visualisations under discussion.
- Annotation. From logging the sample through to publishing the analysis, it is necessary to have annotations that enrich the description of any digital content. This meta-content may apply to data, information or knowledge and depends on agreed interpretations.

- Workflow. To support the process enactment and automation, the system needs descriptions of processes. The scenario illustrates workflow both inside and outside the company.
- Notification. The arrival of new information prompts notifications to users and initiates automatic processing.
- Decision support. The technicians and scientists are provided with relevant information and suggestions for the task at hand.
- Resource reservation. There is a need to ease the process of resource reservation. This applies to experimental equipment, collaboration (the conference), and resource scheduling for the simulation.
- Security. There are authentication, encryption and privacy requirements, with multiple organisations involved, and a requirement for these to be handled with minimal manual intervention.
- Reliability. The systems are appear to be reliable but in practice there may be failures and exception handling at various levels, including the workflow.
- Video. Both live and stored video have a role, especially where the video is enriched by associated temporal data (in this case to aid navigation).
- Smart laboratory. For example, the equipment detects the sample (e.g. by barcode or RFID tag), the scientist may use portable devices for note-taking, and visualisations may be available in the lab.
- Knowledge. Knowledge services are an integral part of the e-Science process. Examples include: finding papers, finding people, finding previous experimental design (these queries may involve inference), annotating the uploaded analysis, configuring the lab to the person.
- Growth. The system should support evolutionary growth as new content and processing techniques become available.
- Scale. The scale of the scientific collaboration increases through the scenario, as does the scale of computation, bandwidth, storage and complexity of relationships between information.

We believe that these requirements will be ubiquitous in e-Science applications conducted or undertaken in a grid context. The rest of this document explores the issues that arise in trying to satisfy these requirements.

1.3 Report Structure

The remainder of this document is organised in the following manner:

Section 2. Service-Oriented Architectures
Justification and presentation of the service-oriented view at the various levels of the infrastructure. This section feeds into sections 4 and 5 in particular.

Section 3. The Data/Computation Layer
Discussion of the distributed computing frameworks that are appropriate for grid computing, informed by existing grid computing activities.

Section 4. The Information Layer
Discussion of the Web, distributed information management research and collaborative information systems that can be exploited in e-Science applications.

Section 5. The Knowledge Layer
Description of the tools and techniques related to knowledge capture, modelling and publication that are pertinent to e-Science applications.

Section 6. **Research agenda**
Summary of the steps that need to be undertaken in order to realise the vision of the Semantic Grid as outlined in this report.

2. A Service-Oriented View

This section expands upon the view of the e-Science infrastructure as a service-oriented architecture in which *entities* provide *services* to one another under various forms of *contract.* Thus, as shown in figure 1.1, the e-Scientist's environment is composed of data/computation services, information services, and knowledge services. However, before we deal with the specifics of each of these different types of service, respectively in sections 3, 4 and 5, it is important to highlight those aspects that are common since this provides the conceptual basis and rationale for what follows. To this end, section 2.1 provides the justification for a *service-oriented* view of the different layers of the e-Science infrastructure. Section 2.2 then addresses the technical ramifications of this choice and outlines the key technical challenges that need to be overcome to make service-oriented grids a reality. The section concludes (section 2.3) with the e-Science scenario of section 1.2 expressed in a service-oriented architecture.

2.1 Justification of a Service-Oriented View

Given the set of desiderata and requirements from section 1.2, a key question in designing and building grid applications is what is the most appropriate conceptual model for the system? The purpose of such a model is to identify the key constituent components (abstractions) and specify how they are related to one another. Such a model is necessary to identify generic grid technologies and to ensure that there can be re-use between different grid applications. Without a conceptual underpinning, grid endeavours will simply be a series of handcrafted and *ad hoc* implementations that represent point solutions.

To this end, an increasingly common way of viewing many large systems (from governments, to businesses, to computer systems) is in terms of the *services* that they provide. Here a service can simply be viewed as an abstract characterization and encapsulation of some content or processing capabilities. For example, potential services in our exemplar scenario could be: the equipment automatically recognising the sample and configuring itself appropriately, the logging of data about a sample in the international database, the setting up of a video to monitor the experiment, the locating of appropriate computational resources to support a run of the High Resolution Analyser, the finding of all scientists who have published work on experiments similar to those uncovered by our e-Scientist, and the analyser raising an alert whenever a particular pattern of results occurs (see section 2.3 for more details). Thus, services can be related to the domain of the Grid, the infrastructure of the computing facility, or the users of the Grid – i.e., at the data/computation layer, at the information layer, or at the knowledge layer. In all of these cases, however, it is assumed that there may be multiple versions of broadly the same service present in the system.

Services do not exist in a vacuum, rather they exist in a particular institutional context. Thus all services have an owner (or set of owners). The owner is the body (individual or institution) that is responsible for offering the service for consumption by others. The owner sets the terms and conditions under which the service can be accessed. Thus, for example, the owner may decide to make the service universally available and free to all on a first-come, first-served basis. Alternatively, the owner may decide to limit access to particular classes of users, to charge a fee for access and to have priority-based access. All options between these two extremes are also possible. It is assumed that in a given system there will be multiple service owners (each representing a different stakeholder) and that a given service owner may offer multiple services. These services may correspond to genuinely different functionality or they may vary in the way that broadly the same functionality is delivered (e.g., there may be a quick and approximate version of the service and one that is more time consuming and accurate).

In offering a service for consumption by others, the owner is hoping that it will indeed attract consumers for the service. These consumers are the entities that decide to try and invoke the service. The purpose for which this invocation is required is not of concern here: it may be for their own private use, it may be to resell onto others, or it may be to combine with other services.

The relationship between service owner and service consumer is codified through a service contract. This contract specifies the terms and conditions under which the owner agrees to provide the service to the consumer. The precise structure of the contract will depend upon the nature of the service and the relationship between the owner and the provider. However examples of relevant attributes include the price for invoking the service, the information the consumer has to provide to the provider, the expected output from the service, an indication about when this output can be expected, and the penalty for failing to deliver according to the contract. Service contracts can either be established by an off-line or an on-line process depending on the prevailing context.

The service owners and service producers interact with one another in a particular environmental context. This environment may be common to all entities in the Grid (meaning that all entities offer their services in an entirely open marketplace). In other cases, however, the environment may be closed and entrance may be controlled (meaning that the entities form a private club).[2] In what follows, a particular environment will be called a *marketplace* and the entity that establishes and runs the marketplace will be termed the *market owner*. The rationale for allowing individual marketplaces to be defined is that they offer the opportunity to embed interactions in an environment that has its own set of rules (both for membership and ongoing operation) and they allow the entities to make stronger assumptions about the parties with which they interact (e.g., the entities may be more trustworthy or cooperative since they are part of the same club). Such marketplaces may be appropriate, for example, if the nature of the domain means that the services are particularly sensitive or valuable. In such cases, the closed nature of the marketplace will enable the entities to interact more freely because of the rules of membership.

To summarise, the key components of a service-oriented architecture are as follows (figure 2.1): service owners (rounded rectangles) that offer services (filled circles) to service consumers (filled triangles) under particular contracts (solid links between producers and consumers). Each owner-consumer interaction takes place in a given

[2] This is analogous to the notion of having a virtual private network overlaid on top of the Internet. The Internet corresponds to the open marketplace in which anybody can participate and the virtual private network corresponds to a closed club that can interact under its own rules.

marketplace (denoted by ovals) whose rules are set by the market owner (filled cross). The market owner may be one of the entities in the marketplace (either a producer or a consumer) or it may be a neutral third party.

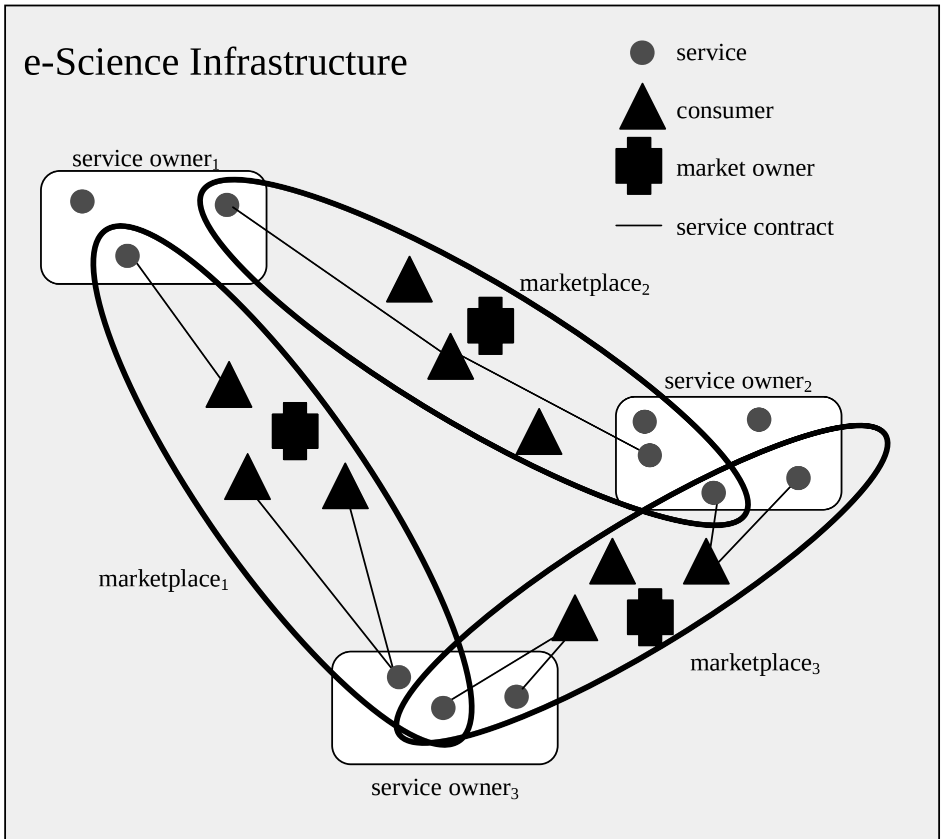

Figure 2.1: Service-oriented architecture: key components

Given the central role played by the notion of a service, it is natural to explain the operation of the system in terms of a *service lifecycle* (figure 2.2). The first step is for a service owner to define a service they wish to make available to others. The reasons for wanting to make a service available may be many and varied – ranging from altruism, through necessity, to commercial benefit. It is envisaged that in a given grid application all three motivations (and many others besides) are likely to be present, although perhaps to varying degrees that are dictated by the nature of the domain. *Service creation* should be seen as an ongoing activity. Thus new services may come into the environment at any time and existing ones may be removed (service decommissioning) at any time. This means the system is in a state of continual flux and never reaches a steady state. Creation is also an activity that can be automated to a greater or less extent. Thus, in some cases, all services may be put together in an entirely manual fashion. In other cases, however, there may be a significant automated component. For example, it may be decided that a number of services should be combined; either to offer a new service (if the services are complementary in nature) or to alter the ownership structure (if the services are similar). In such cases, it may be appropriate to automate the processes of finding appropriate service providers and of getting them to agree to new terms of operation. This dynamic service composition activity is akin to creating a new *virtual organisation*: a number of initially distinct entities can come together, under a set of operating conditions, to form a new entity that offers a new service. This grouping will then stay in place until it is no longer appropriate to remain in this form, whereupon it will disband.

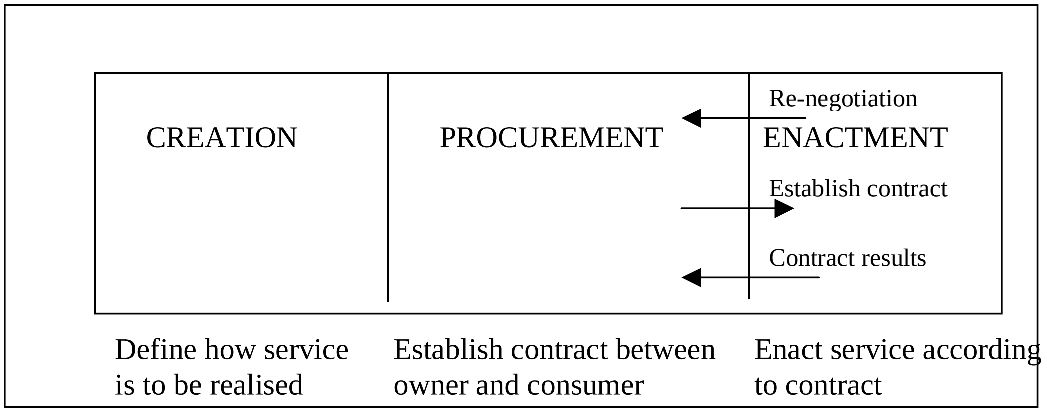
CREATION
PROCUREMENT
ENACTMENT
Re-negotiation
Establish contract
Contract results
Define how service is to be realised
Establish contract between owner and consumer
Enact service according to contract

Figure 2.2: Service lifecycle

The *service creation* process covers three broad types of activity. Firstly, specifying how the service is to be realized by the service owner using an appropriate service description language. These details are not available externally to the service consumer (i.e., they are encapsulated by the service owner). Secondly, specifying the meta-information associated with the service. This indicates the potential ways in which the service can be procured. This meta-information indicates who can access the service and what are the likely contract options for procuring it (see section 4 for more details). Thirdly, making the service available in the appropriate marketplace. This requires appropriate service advertising and registration facilities to be available in the marketplace (see section 4 for more details).

The *service procurement* phase is situated in a particular marketplace and involves a service owner and a service consumer establishing a contract for the enactment of the service according to a particular set of terms and conditions. There are a number of points to note about this process. Firstly, it may fail. That is, for whatever reason, a service owner may be unable or unwilling to provide the service to the consumer. Secondly, in most cases, the service owner and the service consumer will represent different and autonomous stakeholders. Thus the process by which contracts are established will be some form of *negotiation* – since the entities involved need to come to a mutually acceptable agreement on the matter. If the negotiation is successful (i.e., both parties come to an agreement) then the outcome of the procurement is a contract between the service owner and the service consumer. Thirdly, this negotiation may be carried out off-line by the respective service owners or it may be carried out at run-time. In the latter case, the negotiation may be automated to a greater or lesser extent – varying from the system merely automatically flagging the fact that a new service contract needs to be established to automating the entire negotiation process[3].

[3] Automated negotiation technology is now widely used in many e-commerce applications. It encompasses various forms of auctions (a one-to-many form of negotiation) as well as bi-lateral negotiations. Depending on the negotiation protocol that is in place, the negotiation can be concluded

The final stage of the service lifecycle is *service enactment*. Thus, after having established a service contract, the service owner has to undertake the necessary actions in order to fulfil its obligations as specified in the contract. After these actions have been performed, the owner needs to fulfil its reporting obligations to the consumer with respect to the service. This may range from a simple inform indicating that the service has been completed, to reporting back complex content which represents the results of performing the service. The above assumes that the service owner is always able to honour the contracts that it establishes. However, in some cases the owner may not be able to stick to the terms specified in the contract. In such cases, it may have to renegotiate the terms and conditions of the contract; paying any penalties that are due. This enforcement activity is undertaken by the market owner and will be covered by the terms and conditions that the service providers and consumers sign up to when they enter into the marketplace.

Having described the key components of the service-oriented approach, we return to the key system-oriented desiderata noted in section 1.2. From the above discussion, it can be seen that a service-oriented architecture is well suited to grid applications:

- able to store and process huge volumes of content in a timely fashion;
    - *The service-oriented model offers a uniform means of describing and encapsulating activities at all layers in the Grid. This model then needs to be underpinned by the appropriate processing and communication infrastructure to ensure it can deliver the desired performance (see sections 3 and 4 for more details).*

- allow different stakeholders to retain ownership of their own content and processing capabilities, but to allow others access under the appropriate terms and conditions;
    - *Each service owner retains control over the services that they make available to others. They determine how the service is realized and set the policy for accessing the service.*

- allow users to discover, transparently access and process relevant content wherever it may be located in the Grid;
    - *The overall system is simply viewed as a number of service marketplaces. Any physical distribution and access problems are masked via the service interface and the service contract (see section 3 for more details). The marketplace itself has advertisement and brokering mechanisms to ensure appropriate service owners and consumers are put together (see section 4 for more details).*

- allow users to form, maintain and disband communities of practice with restricted membership criteria and rules of operation;
    - *Each community can establish its own marketplace. The marketplace owner defines the conditions that have to be fulfilled before entities can*

in a single round or it may last for many rounds. Thus negotiation need not be a lengthy process; despite the connotation from human interactions that it may be!

enter, defines the rules of interaction for once the marketplace is operational, and enforces the rules through appropriate monitoring.

- allow content to be combined from multiple sources in unpredictable ways according to the users' needs;
    - *It is impossible to a priori predict how the users of a system will want to combine the various services contained within it. Thus services must be able to be composed in flexible ways. This is achieved by negotiation of appropriate contracts. This composition can be done on a one-off basis or may represent a more permanent binding into a new service that is offered on an ongoing basis (as in the establishment of a new virtual organisation).*

- support evolutionary growth as new content and processing techniques become available.
    - *Services represent the unit of extension of the system. Thus as new content or processing techniques become available they are simply represented as new services and placed in a marketplace(s). Also new marketplaces can be added as new communities of practice emerge.*

2.2 Key Technical Challenges

The previous section outlined the service-oriented view of grid architectures. Building upon this, this section identifies the key technical challenges that need to be overcome to make such architectures a reality. To this end, table 2.1 represents the key functionality of the various components of the service-oriented architecture, each of which is then described in more detail in the remainder of this section.

Service Owner	**Service Consumer**	**Marketplace**
Service creation	Service discovery	Owner and consumer registration
Service advertisement		Service registration
Service contract creation	Service contract creation	Policy specification
Service delivery	Service result reception	Policy monitoring and enforcement

Table 2.1: Key functions of the service-oriented architecture components

2.2.1 Service Owners and Consumers as Autonomous Agents

A natural way to conceptualise the service owners and the service consumers are as autonomous agents. Although there is still some debate about exactly what constitutes agenthood, an increasing number of researchers find the following characterisation useful [Wooldridge97]:

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

There are a number of points about this definition that require further explanation. Agents are [Jennings00]: (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 purpose—they have particular objectives (goals) to achieve; (iv) autonomous— they have control both over their internal state and over their own behaviour[4]; (v) capable of exhibiting flexible problem solving behaviour 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 act in anticipation of future goals) .

Thus, each service owner will have one or more agents acting on its behalf. These agents will manage access to the services for which they are responsible and will ensure that the agreed contracts are fulfilled. This latter activity involves the scheduling of local activities according to the available resources and ensuring that the appropriate results from the service are delivered according to the contract in place. Agents will also act on behalf of the service consumers. Depending on the desired degree of automation, this may involve locating appropriate services, agreeing contracts for their provision, and receiving and presenting any received results.

2.2.2 Interacting Agents

Grid applications involve multiple stakeholders interacting with one another in order to procure and deliver services. Underpinning the agents' interactions is the notion that they need to be able to inter-operate in a meaningful way. Such semantic interoperation is difficult to obtain in grids (and all other open systems) because the different agents will typically have their own individual information models. Moreover, the agents may have a different communication language for conveying their own individual terms. Thus, meaningful interaction requires mechanisms by which this basic interoperation can be effected (see sections 4 and 5 for more details).

Once semantic inter-operation has been achieved, the agents can engage in various forms of interaction. These interactions can vary from simple information interchanges, to requests for particular actions to be performed and on to cooperation, coordination and negotiation in order to arrange interdependent activities. In all of these cases, however, there are two points that qualitatively differentiate agent interactions from those that occur in other computational models. Firstly, agent-oriented interactions are conceptualised as taking place at the *knowledge level* [Newell82]. That is, they are conceived in terms of which goals should be followed, at what time, and by whom. Secondly, as agents are flexible problem solvers, operating in an environment over which they have only partial control and observability, interactions need to be handled in a similarly flexible manner. Thus, agents need the computational apparatus to make *run-time* decisions about the nature and scope of

[4] Having control over their own behaviour is one of the characteristics that distinguishes agents from objects. Although objects encapsulate state and behaviour (more accurately behaviour realisation), they fail to encapsulate behaviour activation or action choice. Thus, any object can invoke any publicly accessible method on any other object at any time. Once the method is invoked, the corresponding actions are performed. In this sense, objects are totally obedient to one another and do not have autonomy over their choice of action.

their interactions and to initiate (and respond to) interactions that were not foreseen at design time (cf. the hard-wired engineering of such interactions in extant approaches).

The subsequent discussion details what would be involved if all these interactions were to be automated and performed at run-time. This is clearly the most technically challenging scenario and there are a number of points that need to be made. Firstly, while such automation is technically feasible, in a limited form, using today's technology, this is an area that requires more research to reach the desired degree of sophistication and maturity. Secondly, in some cases, the service owners and consumers may not wish to automate all of these activities since they may wish to retain a degree of human control over these decisions. Thirdly, some contracts and relationships may be set up at design time rather than being established at run-time. This can occur when there are well-known links and dependencies between particular services, owners and consumers.

The nature of the interactions between the agents can be broadly divided into two main camps. Firstly, those that are associated with making service contracts. This will typically be achieved through some form of automated negotiation since the agents are autonomous [Jennings01]. When designing these negotiations, three main issues need to be considered:

- *The Negotiation Protocol*: the set of rules that govern the interaction. This covers the permissible types of participants (e.g. the negotiators and any relevant third parties), the negotiation states (e.g. accepting bids, negotiation closed), the events that cause negotiation states to change (e.g. no more bidders, bid accepted) and the valid actions of the participants in particular states (e.g. which messages can be sent by whom, to whom, at what stage).

- *The Negotiation Object*: the range of issues over which agreement must be reached. At one extreme, the object may contain a single issue (such as price), while on the other hand it may cover hundreds of issues (related to price, quality, timings, penalties, terms and conditions, etc.). Orthogonal to the agreement structure, and determined by the negotiation protocol, is the issue of the types of operation that can be performed on agreements. In the simplest case, the structure and the contents of the agreement are fixed and participants can either accept or reject it (i.e. a take it or leave it offer). At the next level, participants have the flexibility to change the values of the issues in the negotiation object (i.e. they can make counter-proposals to ensure the agreement better fits their negotiation objectives). Finally, participants might be allowed to dynamically alter (by adding or removing issues) the structure of the negotiation object (e.g. a car salesman may offer one year's free insurance in order to clinch the deal).

- *The Agent's Decision Making Models*: the decision-making apparatus the participants employ to act in line with the negotiation protocol in order to achieve their objectives. The sophistication of the model, as well as the range of decisions that have to be made, are influenced by the protocol in place, by the nature of the negotiation object, and by the range of operations that can be performed on it. It can vary from the very simple, to the very complex.

In designing any automated negotiation system the first thing that needs to be established is the protocol to be used. In this context, this will be determined by the market owner. Here the main consideration is the nature of the negotiation. If it is a one-to-many negotiation (i.e., one buyer and many sellers or one seller and many buyers) then the protocol will typically be a form of auction. Although there are thousands of different permutations of auction, four main ones are typically used. These are: English, Dutch, Vickrey, and First-Price Sealed Bid. In an English auction, the auctioneer begins with the lowest acceptable price and bidders are free to raise their bids successively until there are no more offers to raise the bid. The winning bidder is the one with the highest bid. The Dutch auction is the converse of the English one; the auctioneer calls for an initial high price, which is then lowered progressively until there is an offer from a bidder to claim the item. In the first-priced sealed bid, each bidder submits their offer for the item independently without any knowledge of the other bids. The highest bidder gets the item and they pay a price equal to their bid amount. Finally, a Vickrey auction is similar to a first-price sealed bid auction, but the item is awarded to the highest bidder at a price equal to the second highest bid. More complex forms of auctions exist to deal with the cases in which there are multiple buyers and sellers that wish to trade (these are called double auctions) and with cases in which agents wish to purchase multiple interrelated goods at the same time (these are called combinatorial auctions). If it is a one-to-one negotiation (one buyer and one seller) then a form of heuristic model is needed (e.g. [Faratin99; Kraus01]). These models vary depending upon the nature of the negotiation protocol and, in general, are less well developed than those for auctions.

Having determined the protocol, the next step is to determine the nature of the contract that needs to be established. This will typically vary from application to application and again it is something that is set by the market owner. Given these two, the final step is to determine the agent’s reasoning model. This can vary from the very simple (bidding truthfully) to the very complex (involving reasoning about the likely number and nature of the other bidders).

The second main type of interaction is when a number of agents decide to come together to form a new virtual organisation. This involves determining the participants of the coalition and determining their various roles and responsibilities in this new organisational structure. Again this is typically an activity that will involve negotiation between the participants since they need to come to a mutually acceptable agreement about the division of labour and responsibilities. Here there are a number of techniques and algorithms that can be employed to address the coalition formation process [Sandholm00; Shehory98] although this area requires more research to deal with the envisaged scale of grid applications.

2.2.3 Marketplace Structures

Marketplaces should be able to be established by any agent(s) in the system (including a service owner, a service consumer or a neutral third party). The entity which establishes the marketplace is here termed the market owner. The owner is responsible for setting up, advertising, controlling and disbanding the marketplace. In order to establish a marketplace, the owner needs a representation scheme for describing the various entities that are allowed to participate in the marketplace (terms of entry), a means of describing how the various allowable entities are allowed to interact with one another in the context of the marketplace, and what monitoring mechanisms (if any) are to be put in place to ensure the marketplace's rules are adhered to.

2.3 A Service-Oriented View of the Scenario

Having reviewed the service-oriented approach we will now apply this analysis and framework to the scenario described in section 1.2.

The first marketplace is that connected with the scientist's own lab. This marketplace has agents to represent the humans involved in the experiment, thus there is a *scientist agent* (SA) and a *technician agent* (TA). These are responsible for interacting with the scientist and the technician, respectively, and then for enacting their instructions in the Grid. These agents can be viewed as the computational proxies of the humans they represent – endowed with their personalised information about their owner's preferences and objectives. These personal agents need to interact with other (artificial) agents in the marketplace in order to achieve their objectives. These other agents include an *analyser agent* (AA) (that is responsible for managing access to the analyser itself), the *analyser database agent* (ADA) (that is responsible for managing access to the database containing information about the analyser), and the *high resolution analyser agent* (HRAA) (that is responsible for managing access to the high resolution analyser). There is also an *interest notification agent* (INA) (that is responsible for recording which scientists in the lab are interested in which types of results and for notifying them when appropriate results are generated) and an *experimental results agent* (ERA) (that can discover similar analyses of data or when similar experimental configurations have been used in the past). The services provided by these agents are summarised in table 2.2.

Agent	Services Offered	Services Consumed By
Scientist Agent (SA)	`resultAlert`	Scientist
	`reportAlert`	Scientist
Technician Agent (TA)	`MonitorAnalysis`	Technician
Analyser Agent (AA)	`configureParameters`	ADA
	`runSample`	ADA
Analyser Database Agent (ADA)	`logSample`	Technician
	`setAnalysisConfiguration`	Technician
	`bookSlot`	TA
	`recordAnalysis`	AA
High Resolution Analyser Agent (HRAA)	`bookSlot`	SA
	`configureParameters`	Scientist
	`runAnalysis`	Scientist
	`videoAnalysis`	Scientist, Technician
	`monitorAnalysis`	Technician
	`reportResults`	SA
	`replayExperiment`	Scientist
	`suggestRelatedConfigurations`	Scientist
Interest Notification Agent (INA)	`registerInterest`	Scientists, Technicians
	`notifyInterestedParties`	ADA
	`findInterestedParties`	Scientist
Experimental Results Agent (ERA)	`FindSimilarExperiments`	HRAA

Table 2.2: Services in the scientist's lab marketplace

The operation of this marketplace is as follows. The technician uses the `logSample` service to record data about the sample when it arrives and the `setAnalysisConfiguration` service to set the appropriate parameters for the forthcoming experiment. The technician then instructs the TA to book a slot on the analyser using the `bookSlot` service. At the appropriate time, the ADA informs the AA of the settings that it should adopt (via the `configureParameters` service) and that it should now run the experiment (via the `runSample` service). As part of the contract for the `runSample` service, the AA informs the ADA of the results of the experiment and these are logged along with the appropriate experimental settings (using the `recordAnalysis` service). Upon receipt of these results, the ADA informs the INA of them. The INA then disseminates the results (via the `notifyInterestedParties` service) to scientists who have registered an interested in results of that kind (achieved via the `registerInterest` service).

When interesting results are received, the SA alerts the scientist (via the `resultAlert` service). The scientist then examines the results and decides that they are of interest and that further analysis is need. The scientist then instructs the SA to make a booking on the High Resolution Analyser (via the `bookSlot` service). When the booking is made, the HRAA volunteers information to the scientist about the configurations of similar experiments that have previously been run (via the `suggestRelatedConfigurations` service). Using this information, the scientist sets the appropriate configurations (via the `configureParameters` service). At the appropriate time, the experiment is started (via the `runAnalysis` service). As part of the contract for this service, the experiment is videoed (via the `videoAnalysis` service), monitoring information is sent to the technician (via the `monitorAnalysis` service) and a report is prepared and sent to the SA (via the `reportResults` service). In preparing this report, the HRAA interacts with the ERA to discover if related experiments and results have already been undertaken (achieved via the `findSimilarExperiments` service).

The scientist is alerted to the report by the SA (via the `reportAlert` service). The scientist decides the results may be interesting and decides to replay some of the key segments of the video (via the `replayExperiment` service). The scientist decides the results are indeed interesting and so asks for relevant publications and details of scientists who have published on this topic. This latter activity is likely to be provided through an external marketplace that provides this service for the wider community (see table 2.4). In such a marketplace, there may be multiple *Paper Repository Agents* that offer the same broad service (`findRelatedPapers` and `findRelatedAuthors`) but to varying degrees of quality, coverage, and timeliness.

Armed with all this information, the scientist decides that the results should be discussed within the wider organisation context. This involves interacting in the Scientist's Organisation Marketplace. The agents involved in this marketplace are the *research meeting convener agent* (RMCA) (responsible for organising research meetings) and the various *scientist agents* that represent the relevant scientists. The services provided by these agents are given in table 2.3. The RMCA is responsible for determining when research meetings should take place, this is achieved via the `arrangeMeeting` service through interaction with the SAs of the scientists involved. The scientist requests a slot to discuss the latest experimental findings (via the `setAgenda` service) and provides the appropriate data for discussion to the RMCA that disseminates it to the SA's of the relevant participants (via the disseminateInformation service). As a consequence of the meeting, it is decided that the results are appropriate for dissemination into the scientific community at large.

Agent	Services Offered	Service Consumed By
Research Meeting Convener Agent (RMCA)	arrangeMeeting setAgenda disseminateInformation	SAs Scientist SAs
Scientist Agent (SA)	arrangeMeeting receiveInformation	RMCA RMCA

Table 2.3: Services in the scientist's organisation marketplace

The general scientific community is represented by a series of distinct marketplaces that are each responsible for different aspects of the scientific process. As decided upon at the organisation's meeting, the sample data is logged in the appropriate international database (using the logSample service). This database has an attached notification service at which individual scientists can register their interests in particular types of data (via the registerInterests service). Scientists will then be informed, via their SA, when new relevant data is posted (via the disseminateInformation service).

	Services Offered	Services Consumed By
International Sample Database Agent (ISDA)	logSample registerInterests disseminateInformation	Scientist Scientist SAs
Paper Repository Agent (PRA)	findRelatedPapers findRelatedAuthors	SAs SAs
Scientist Agent (SA)	receiveRelevantData arrangeSimulation	Scientist Scientist
Simulation Provider Agent (SPA)	offerSimulationResource utiliseSimulationResource	SA SA
Problem Solving Environment Agent (PSEA)	whatSimulationTools simulationSettingInfo analyseResults	Scientist Scientist Scientist

Table 2.4: Services in the general scientific community marketplace

One of the scientists who receives notification of the new results believes that they should be investigated further by undertaking a new round of simulations. The scientist instructs the SA to arrange for particular simulations to be arranged. The SA enters a marketplace where providers of processing capabilities offer their resources (via the offerSimulationResource service). The SA will arrange for the appropriate amount of resource to be made available at the desired time such that the simulations can be run. Once these contracts have been established, the SA will invoke the simulation (via the utiliseSimulationResource service). During the course of these simulations, the scientist will make use of the Problem Solving Environment Agent (PSEA) to assist in the tasks of determining what simulation tools to exploit (via the whatSimulationTools service), setting the simulation parameters appropriately for these tools (via the `simulationSettingInfo` service), and analysing the results (via the `analyseResults` service).

This then characterises our scenario as an active marketplace of agents offering and consuming services. As already indicated, we do not expect that this complete set of interactions will be dealt with seamlessly by computational agents in the near future. However, it provides a level of abstraction and defines capabilities that we claim it is important to aspire to if the full potential of the Semantic Grid is to be realised.

3. The Data-Computation Layer

As soon as computers are interconnected and communicating we have a distributed system, and the issues in designing, building and deploying distributed computer systems have now been explored over many years. This section considers e-Science infrastructure from a distributed systems perspective. First it positions the Grid within the bigger picture of distributed computing, asking whether it is subsumed by current solutions. Then we look in more detail at the requirements and currently deployed technologies in order to identify issues for the next generation of the infrastructure. Since much of the grid computing development has addressed the data-computation layer, this section particularly draws upon the work of that community.

Our discussion of the data/computation layer makes certain assumptions of the underlying network fabric, which are not addressed in detail here:

- *Performance*. On the whole the requirements of e-Science are not substantially different in nature from other distributed applications. In addition to bandwidth, perhaps the two most stringent requirements are the latency requirements of some parallel algorithms, and the multimedia quality of service requirements imposed by visualisation and the multicast videoconferencing facility known as the Access Grid (see section 4.2.2).

- *Management*. Aside from the performance of the fabric, there may be implications for network resource management, for example the automation of network resource reservation and handling of different types of service. It may also be necessary for the data/computation layer to have access to information about the dynamic state of the fabric.

- *Security*. We assume that the network meets the application’s security requirements, for example by use of firewalls, secure internet protocols, authentication and access control. This also has network management implications, particularly where grid applications cross organisational boundaries.

Furthermore we do not study storage devices nor particular processing solutions. With these caveats, the remainder of this section is organised as follows. Section 3.1 deals with grid computing as a distributed system, section 3.2 with the requirements at the data-computational layer, section 3.3 with technologies that are being used at this layer, section 3.4 with grid portals, section 3.5 with grid deployments, and section 3.6 with the open research issues.

3.1 Grid Computing as a Distributed System

Fundamentally, distribution introduces notions of concurrency and locality. The traditional view of a computation involves the data, program and computational state (memory, processor registers) coinciding at one processing node. There are many techniques for achieving this in a distributed system and the following are illustrative examples:

- Remote Procedure Calls. The code is installed in advance on the ‘server’. The client sends parameters, a computation is invoked, and a result is returned. The model is essentially synchronous. This approach is exemplified by the Distributed Computing Environment (DCE) from the Open Software Foundation (now Open Group).
- Services[5]. Again the code is server-side and the computation runs continuously (a UNIX daemon or Windows service), clients connect and there is a bi-directional interaction according to a high level communications protocol.
- Distributed object systems. Within the object-oriented paradigm, remote method invocation replaces remote procedure calls, usually with support for object request brokering.
- Database queries. The database is running on the server, a client sends a small program (i.e. the query) and this is executed on the server, data is exchanged.
- Client-side scripting. The client downloads the script from the server and executes it locally in a constrained environment, such as a Web browser.
- Message passing. This type of paradigm encapsulates distributed ‘parallel’ programming such as MPICH or PVM (note additionally that process invocation may occur through techniques such as a remote shell). It also includes message queuing systems.
- Peer-to-peer. Those devices that traditionally act as clients (the office workstation, the home computer) can also offer services, including computational services; i.e. computation at the ‘edge’ of the network. In principle such systems can be highly decentralised, promoting scalability and avoiding single points of failure.
- Web-based computing. This involves the use of the Web infrastructure as a fabric for distributed applications, and closely resembles Unix services and remote procedure calls.

Each of the above models is in widespread use and when taken together they illustrate the current practice in distributed systems deployment. It seems likely that grid computing will not be based on just one of these technologies: they are the legacy with which the infrastructure must interoperate, and although proponents of any one of the above might argue that theirs is a universal solution, no dominant solution is apparent in current practice. Of course they are closely inter-related and any system could be expressed in multiple ways – different instances of a conceptual model.

So the key question to ask in this context is will this change as grid computing evolves? In some respects the same distributed systems issues continue to apply. For example, although wide area networks now operate at LAN speeds, the scale of the wide area has grown too. Availability of very high speed networking might de-emphasise the need for locality, but the global geographical scale of the Grid re-emphasises it; e.g. local memory access must still be more efficient as the latency over the wider area is fundamentally constrained by the speed of light. Data locality continues to be key to application efficiency and performance. So although local computational resources may use a variety of architectures (e.g. tightly coupled multiprocessors), over the wide area the grid architecture is a classical distributed system.

[5] This notion of services is different from that described in section 2, although it can be regarded as a particular technical implementation of these ideas.

What will certainly change is *scale*, and with it *heterogeneity*. These are both perennial challenges in distributed systems, and both are critical in the e-Science context. In fact, in many ways the “e-” prefix denotes precisely these things, as characterised by phrases such as ‘distributed global collaborations’, ‘very large data collections’, and ‘tera-scale computing resources’. With regards to scaling up the number of nodes, difficulties are experienced with the computational and intellectual complexity of the programs, with configuration of the systems, and with overcoming increased likelihood of failure. Similarly, heterogeneity is inherent – as any cluster manager knows, their only truly homogeneous cluster is their first one! Furthermore, increasing scale also involves crossing an increasing number of organisational boundaries, which magnifies the need to address authentication and trust issues.

There is a strong sense of *automation* in distributed systems; for example, when humans cannot deal with the scale but delegate to processes on the distributed system to do so (e.g. through scripting), which leads to autonomy within the systems. This implies a need for coordination, which, in turn, needs to be specified programmatically at various levels – including process descriptions. Similarly the increased likelihood of failure implies a need for automatic recovery from failure: configuration and repair cannot remain manual tasks.

The inherent heterogeneity must also be handled automatically. Systems use varying standards and system APIs, resulting in the need to port services and applications to the plethora of computer systems used in a grid environment. Agreed interchange formats help in general, because n converters are needed to enable n components to interoperate via one standard, as opposed to n^2 converters for them to interoperate with each other.

To introduce the matching of e-Science requirements to distributed systems solutions, the next three subsections describe the three most prevalent distributed systems solutions with broad claims and significant deployment, considering whether each might meet the needs of the e-Science infrastructure. These systems are: distributed object systems, web-based computing and peer-to-peer solutions. Having positioned grid computing in the general context of distributed systems, we then turn to the specific requirements, technologies, and exemplars of the data-computational layer.

3.1.1 Distributed Object Systems

The Common Object Request Broker Architecture (CORBA) is an open distributed object-computing infrastructure being standardised by the Object Management Group [OMG]. CORBA automates many common network programming tasks such as object registration, location, and activation; request demultiplexing; framing and error-handling; parameter marshalling and demarshalling; and operation dispatching. Although CORBA provides a rich set of services, it does not contain the grid level allocation and scheduling services found in Globus (see section 3.3.1), however, it is possible to integrate CORBA with the Grid.

The OMG has been quick to demonstrate the role of CORBA in the grid infrastructure; e.g. through the ‘Software Services Grid Workshop’ held in 2001. Apart from providing a well-established set of technologies that can be applied to e-Science, CORBA is also a candidate for a higher level conceptual model. It is language-neutral and targeted to provide benefits on the enterprise scale, and is closely associated with the Unified Modelling Language (UML).

One of the concerns about CORBA is reflected by the evidence of intranet rather than internet deployment, indicating difficulty crossing organisational boundaries; e.g. operation through firewalls. Furthermore, realtime and multimedia support was not part of the original design.

While CORBA provides a higher layer model and standards to deal with heterogeneity, Java provides a single implementation framework for realising distributed object systems. To a certain extent the Java Virtual Machine (JVM) with Java-based applications and services are overcoming the problems associated with heterogeneous systems, providing portable programs and a distributed object model through remote method invocation (RMI). Where legacy code needs to be integrated it can be ‘wrapped’ by Java code.

However, the use of Java in itself has its drawbacks, the main one being computational speed. This and other problems associated with Java (e.g. numerics and concurrency) are being addressed by the likes of the Java Grande Forum (a ‘Grande Application’ is ‘any application, scientific or industrial, that requires a large number of computing resources, such as those found on the Internet, to solve one or more problems’) [JavaGrande]. Java has also been chosen for UNICORE (see Section 3.5.3). Thus what is lost in computational speed might be gained in terms of software development and maintenance times when taking a broader view of the engineering of grid applications.

3.1.2 The Web as an Infrastructure for Distributed Applications

Considerable imagination has been exercised in a variety of systems to overcome the Web’s architectural limitations in order to deliver applications and services. The motivation for this is often pragmatic, seeking ways to piggyback off such a pervasive infrastructure with so much software and infrastructure support. However the web infrastructure is not the ideal infrastructure in every case. There is a danger that e-Science infrastructure could fall into this trap, hence it is important to revisit the architecture.

The major architectural limitation is that information flow is always initiated by the client. If an application is ever to receive information spontaneously from an HTTP server then a role reversal is required, with the application accepting incoming HTTP requests. For this reason, more and more applications now also include HTTP server functionality (*cf* peer-to-peer, see below). Furthermore, HTTP is designed around relatively short transactions rather than the longer-running computations. Finally we note that typical web transactions involve a far smaller number of machines that grid computations.

When the Web is used as an infrastructure for distributed applications, information is exchanged between programs, rather than being presented for a human reader. This has been facilitated by XML (and the XML family or recommendations from W3C) as the standard intermediate format for information exchanged in process-to-process communication. XML is not intended for human readership – that it is human-readable text is merely a practical convenience for developers.

Given its ubiquitous nature as an information exchange medium, there is further discussion of the web infrastructure in Section 4.

3.1.3 Peer-to-Peer Computing

One very plausible approach to address the concerns of scalability can be described as *decentralisation*, though this is not a simple solution. The traditional client-server model can be a performance bottleneck and a single point of failure, but is still prevalent because decentralisation brings its own challenges.

Peer-to-Peer (P2P) computing [Clark01a] (as implemented, for example, by Napster [Napster], Gnutella [Gnutella], Freenet [Clarke01b] and JXTA [JXTA]) and Internet computing (as implemented, for example, by the SETI@home [SETI], Parabon [Parabon], and Entropia systems [Entropia]) are examples of the more general computational structures that are taking advantage of globally distributed resources. In P2P computing, machines share data and resources, such as spare computing cycles and storage capacity, via the Internet or private networks. Machines can also communicate directly and manage computing tasks without using central servers. This permits P2P computing to scale more effectively than traditional client-server systems that must expand a server's infrastructure to expand, and this 'clients as servers' decentralisation is attractive with respect to scalability and fault tolerance for the reasons discussed above.

However, there are some obstacles to P2P computing:

- PCs and workstations used in complex P2P applications will require more computing power to carry the communications and security overhead that servers would otherwise handle.
- Security is an issue as P2P applications give computers access to other machines' resources (memory, hard drives, etc). Downloading files from other computers makes the systems vulnerable to viruses. For example, Gnutella users were exposed to the VBS_GNUTELWORM virus. Another issue is the ability to authenticate the identity of the machines with their peers.
- P2P systems have to cope with heterogeneous resources, such as computers using a variety of operating systems and networking components. Technologies such as Java and XML will help overcome some of these interoperability problems.
- One of the biggest challenges with P2P computing is enabling devices to find one another in a computing paradigm that lacks a central point of control. P2P computing needs the ability to find resources and services from a potentially huge number of globally based decentralised peers.

A number of peer-to-peer storage systems are being developed within the research community. In the grid context these raise interesting issues of security and anonymity:

- The Federated, Available, and Reliable Storage for an Incompletely Trusted Environment (FARSITE) serverless distributed file system [Bolosky00].

- OceanStore, a global persistent data store which aims to provide a 'consistent, highly-available, and durable storage utility atop an infrastructure comprised of untrusted servers' and scale to very large numbers of users. [Kubiatowicz00; Zhuang01]
- The Self-Certifying File System (SFS), a network file system that aims to provide strong security over untrusted networks without significant performance costs [Fu00].
- PAST is a 'large-scale peer-to-peer storage utility' that aims to provide high availability, scalability, and anonymity. Documents are immutable and the identities of content owners, readers and storage providers are protected. [Druschel01]

3.2 Data-Computational Layer Requirements

The emphasis of the early efforts in grid computing was driven by the need to link a number of US national supercomputing centres. The I-WAY project, a forerunner of Globus [Foster97a], successfully achieved this goal. Today the grid infrastructure is capable of binding together more than just a few specialised supercomputing centres. A number of key enablers have helped make the Grid more ubiquitous, including the take up of high bandwidth network technologies and adoption of standards, allowing the Grid to be viewed as a viable distributed infrastructure on a global scale that can potentially support diverse applications.

The previous section has shown that there are three main issues that characterise computational and data grids:

- Heterogeneity: a grid involves a multiplicity of resources that are heterogeneous in nature and might span numerous administrative domains across a potentially global expanse.
- Scalability: a grid might grow from few resources to millions. This raises the problem of potential performance degradation as the size of a grid increases. Consequently, applications that require a large number of geographically located resources must be designed to be latency tolerant and exploit the locality of accessed resources.
- Adaptability: in a grid, a resource failure is the rule, not the exception. In fact, with so many resources in a grid, the probability of some resource failing is naturally high. Resource managers or applications must tailor their behaviour dynamically so that they can extract the maximum performance from the available resources and services.

At this layer, the main concern is ensuring that all the globally distributed resources are accessible under the terms and conditions specified by their respective service owners. This layer can consist of all manner of networked resources ranging from computers and mass storage devices to databases and special scientific instruments. These need to be organised so that they provide resource-independent and application-independent services. Examples of these services are an information service that provides uniform access to information about the structure and state of grid resources or a security service with mechanisms for authentication and authorization that can establish a user's identity and create various user credentials.

Against this background, the following are the main design features required at this level of the Grid:

- Administrative Hierarchy – An administrative hierarchy is the way that each grid environment divides itself up to cope with a potentially global extent. The administrative hierarchy, for example, determines how administrative information flows through the Grid.
- Communication Services – The communication needs of applications using a grid environment are diverse, ranging from reliable point-to-point to unreliable multicast. The communications infrastructure needs to support protocols that are used for bulk-data transport, streaming data, group communications, and those used by distributed objects. The network services used also provide the Grid with important QoS parameters such as latency, bandwidth, reliability, fault-tolerance, and jitter control.
- Information Services – A grid is a dynamic environment where the location and type of services available are constantly changing. A major goal is to make all resources accessible to any process in the system, without regard to the relative location of the resource user. It is necessary to provide mechanisms to enable a rich environment in which information about the Grid is reliably and easily obtained by those services requesting the information. The grid information (registration and directory) services provide the mechanisms for registering and obtaining information about the structure, resources, services, and status and nature of the environment.
- Naming Services – In a grid, like in any other distributed system, names are used to refer to a wide variety of objects such as computers, services, or data. The naming service provides a uniform name space across the complete distributed environment. Typical naming services are provided by the international X.500 naming scheme or DNS (the Internet's scheme).
- Distributed File Systems and Caching – Distributed applications, more often than not, require access to files distributed among many servers. A distributed file system is therefore a key component in a distributed system. From an applications point of view it is important that a distributed file system can provide a uniform global namespace, support a range of file I/O protocols, require little or no program modification, and provide means that enable performance optimisations to be implemented (such as the usage of caches).
- Security and Authorisation – Any distributed system involves all four aspects of security: confidentiality, integrity, authentication and accountability. Security within a grid environment is a complex issue requiring diverse resources autonomously administered to interact in a manner that does not impact the usability of the resources and that does not introduce security holes/lapses in individual systems or the environments as a whole. A security infrastructure is key to the success or failure of a grid environment.
- System Status and Fault Tolerance – To provide a reliable and robust environment it is important that a means of monitoring resources and applications is provided. To accomplish this, tools that monitor resources and applications need to be deployed.
- Resource Management and Scheduling – The management of processor time, memory, network, storage, and other components in a grid is clearly important. The overall aim is to efficiently and effectively schedule the applications that need

to utilize the available resources in the distributed environment. From a user's point of view, resource management and scheduling should be transparent and their interaction with it should be confined to application submission. It is important in a grid that a resource management and scheduling service can interact with those that may be installed locally.

- User and Administrative GUI – The interfaces to the services and resources available should be intuitive and easy to use as well as being heterogeneous in nature. Typically user and administrative access to grid applications and services are web based interfaces.

3.3 Technologies for the Data-Computational Layer

There are growing numbers of Grid-related projects, dealing with areas such as infrastructure, key services, collaborations, specific applications and domain portals. Here we identify some of the most significant to date.

3.3.1 Globus

Globus [Globus] [Foster97b] provides a software infrastructure that enables applications to handle distributed, heterogeneous computing resources as a single virtual machine. The Globus project is a U.S. multi-institutional research effort that seeks to enable the construction of computational grids. A computational grid, in this context, is a hardware and software infrastructure that provides dependable, consistent, and pervasive access to high-end computational capabilities, despite the geographical distribution of both resources and users. A central element of the Globus system is the Globus Toolkit, which defines the basic services and capabilities required to construct a computational grid. The toolkit consists of a set of components that implement basic services, such as security, resource location, resource management, and communications.

It is necessary for computational grids to support a wide variety of applications and programming paradigms. Consequently, rather than providing a uniform programming model, such as the object-oriented model, the Globus Toolkit provides a bag of services which developers of specific tools or applications can use to meet their own particular needs. This methodology is only possible when the services are distinct and have well-defined interfaces (APIs) that can be incorporated into applications or tools in an incremental fashion.

Globus is constructed as a layered architecture in which high-level global services are built upon essential low-level core local services. The Globus Toolkit is modular, and an application can exploit Globus features, such as resource management or information infrastructure, without using the Globus communication libraries. The Globus Toolkit currently consists of the following (the precise set depends on Globus version):

- An HTTP-based ‘Globus Toolkit Resource Allocation Manager’ (GRAM) protocol is used for allocation of computational resources and for monitoring and control of computation on those resources.

- An extended version of the File Transfer Protocol, GridFTP, is used for data access; extensions include use of connectivity layer security protocols, partial file access, and management of parallelism for high-speed transfers.
- Authentication and related security services (GSI)
- Distributed access to structure and state information that is based on the Lightweight Directory Access Protocol (LDAP). This service is used to define a standard resource information protocol and associated information model.
- Monitoring of health and status of system components (HBM)
- Remote access to data via sequential and parallel interfaces (GASS) including an interface to GridFTP.
- Construction, caching, and location of executables (GEM)
- Advanced Resource Reservation and Allocation (GARA)

3.3.2 Legion

Legion [Legion] [Grimshaw97] is an object-based ‘metasystem’, developed at the University of Virginia. Legion provides the software infrastructure so that a system of heterogeneous, geographically distributed, high performance machines can interact seamlessly. Legion attempts to provide users, at their workstations, with a single coherent virtual machine.

Legion takes a different approach to Globus to providing to a grid environment: it encapsulates all of its components as objects. The methodology used has all the normal advantages of an object-oriented approach, such as data abstraction, encapsulation, inheritance, and polymorphism. It can be argued that many aspects of this object-oriented approach potentially make it ideal for designing and implementing a complex environment such as a metacomputer. However, using an object-oriented methodology does not come without a raft of problems, many of which are tied-up with the need for Legion to interact with legacy applications and services.

Legion defines the API to a set of core objects that support the basic services needed by the metasystem. The Legion system has the following set of core object types:

- Classes and Metaclasses – Classes can be considered managers and policy makers. Metaclasses are classes of classes.
- Host objects – Host objects are abstractions of processing resources; they may represent a single processor or multiple hosts and processors.
- Vault objects – Vault objects represents persistent storage, but only for the purpose of maintaining the state of object persistent representation.
- Implementation Objects and Caches – Implementation objects hide the storage details of object implementations and can be thought of as equivalent to an executable in UNIX.
- Binding Agents – A binding agent maps object IDs to physical addressees.
- Context objects and Context spaces – Context objects map context names to Legion object IDs, allowing users to name objects with arbitrary-length string names.

3.3.3 WebFlow

WebFlow [Akarsu98] [Haupt99] is a computational extension of the web model that can act as a framework for wide-area distributed computing and metacomputing. The main goal of the WebFlow design was to build a seamless framework for publishing and reusing computational modules on the Web, so that end users, via a web browser, can engage in composing distributed applications using WebFlow modules as visual components and editors as visual authoring tools. WebFlow has a three-tier Java-based architecture that can be considered a visual dataflow system. The front-end uses applets for authoring, visualization, and control of the environment. WebFlow uses a servlet-based middleware layer to manage and interact with backend modules such as legacy codes for databases or high performance simulations. WebFlow is analogous to the Web; web pages can be compared to WebFlow modules and hyperlinks that connect web pages to inter-modular dataflow channels. WebFlow content developers build and publish modules by attaching them to web servers. Application integrators use visual tools to link outputs of the source modules with inputs of the destination modules, thereby forming distributed computational graphs (or compute-webs) and publishing them as composite WebFlow modules. A user activates these compute-webs by clicking suitable hyperlinks, or customizing the computation either in terms of available parameters or by employing some high-level commodity tools for visual graph authoring. The high performance backend tier is implemented using the Globus toolkit:

- The Metacomputing Directory Services (MDS) is used to map and identify resources.
- The Globus Resource Allocation Manager (GRAM) is used to allocate resources.
- The Global Access to Secondary Storage (GASS) is used for a high performance data transfer.

The current WebFlow system is based on a mesh of Java-enhanced web servers (Apache), running servlets that manage and coordinate distributed computation. This management infrastructure is implemented by three servlets: session manager, module manager, and connection manager. These servlets use URL addresses and can offer dynamic information about their services and current state. Each management servlet can communicate with others via sockets. The servlets are persistent and application-independent. Future implementations of WebFlow will use emerging standards for distributed objects and take advantage of commercial technologies, such as CORBA, as the base distributed object model.

WebFlow takes a different approach to both Globus and Legion. It is implemented in a hybrid manner using a three-tier architecture that encompasses both the Web and third party backend services. This approach has a number of advantages, including the ability to plug-in to a diverse set of backend services. For example, many of these services are currently supplied by the Globus toolkit, but they could be replaced with components from CORBA or Legion.

3.3.4 Nimrod/G Resource Broker and GRACE

Nimrod [Nimrod] is a tool designed to aid researchers undertaking parametric studies on a variety of computing platforms. In Nimrod, a typical parametric study is one that requires the same sequential application be executed numerous times with different input data sets, until some problem space has been fully explored.

Nimrod provides a declarative parametric modelling language for expressing an experiment. Domain experts can create a plan for a parametric computation (task farming) and use the Nimrod runtime system to submit, run, and collect the results from multiple computers. Nimrod has been used to run applications ranging from Bio-informatics and Operational Research, to the simulation of business processes.

A new system called Nimrod/G [Buyya00b] uses the Globus middleware services for dynamic resource discovery and dispatching jobs over Grids. The user need not worry about the way in which the complete experiment is set up, data or executable staging, or management. The user can also set the deadline by which the results are needed and the Nimrod/G broker tries to find the optimal resources available and use them so that the user deadline is met or the cost of computation is kept to a minimum.

The current focus of the Nimrod/G project team is on the use of economic theories (as discussed in section 2.1) in grid resource management and scheduling as part of a new framework called GRACE (Grid Architecture for Computational Economy) [Buyya00a]. The components that make up GRACE include global scheduler (broker), bid-manager, directory server, and bid-server working interacting with grid middleware. The GRACE infrastructure APIs that grid tools and applications programmers can use to develop software support the computational economy. Grid resource brokers, such as Nimrod/G, uses GRACE services to dynamically trade with resource owners to select those resources that offer optimal user cost or timelines criteria.

3.3.5 Jini and RMI

Jini [Jini] is designed to provide a software infrastructure that can form a distributed computing environment that offers network plug and play. A collection of Jini-enabled processes constitutes a Jini community – a collection of clients and services all communicating by the Jini protocols. In Jini, applications will normally be written in Java and communicate using the Java Remote Method Invocation (RMI) mechanism. Even though Jini is written in pure Java, neither Jini clients nor services are constrained to be pure Java. They may include native code, acting as wrappers around non-Java objects, or even be written in some other language altogether. This enables a Jini community to extend beyond the normal Java framework and link services and clients from a variety of sources.

More fundamentally, Jini is primarily concerned with communications between devices (not what devices do). The abstraction is the service and an interface that defines a service. The actual implementation of the service can be in hardware, software, or both. Services in a Jini community are mutually aware and the size of a community is generally considered that of a workgroup. A community's lookup service (LUS) can be exported to other communities, thus providing interaction between two or more isolated communities.

In Jini a device or software service can be connected to a network and can announce its presence. Clients that wish to use such a service can then locate it and call it to perform tasks. Jini is built on RMI, which introduces some constraints**.** Furthermore, Jini is not a distributed operating system, as an operating system provides services such as file access, processor scheduling, and user logins.

The five key concepts of Jini are:

- Lookup – search for a service and download the code needed to access it;
- Discovery – spontaneously find a community and join;
- Leasing – time-bounded access to a service;
- Remote Events – service A notifies service B of A's state change. Lookup can notify all services of a new service;
- Transactions – used to ensure that a system's distributed state stays consistent.

3.3.6 The Common Component Architecture Forum

The Common Component Architecture Forum [CCA] is attempting to define a minimal set of standard features that a high performance component framework would need to provide, or can expect, in order to be able to use components developed within different frameworks. Such a standard will promote interoperability between components developed by different teams across different institutions.

The idea of using component frameworks to deal with the complexity of developing interdisciplinary high performance computing applications is becoming increasingly popular. Such systems enable programmers to accelerate project development by introducing higher-level abstractions and allowing code reusability. They also provide clearly defined component interfaces, which facilitate the task of team interaction. These potential benefits have encouraged research groups within a number of laboratories and universities to develop, and experiment with prototype systems. This has led to some fragmentation and there is a need for interoperability standards.

3.3.7 Batch/Resource Scheduling

There are three freely available systems whose primary focus is batching and resource scheduling:

- Condor [Condor] is a software package for executing batch jobs on a variety of UNIX platforms, in particular those that would otherwise be idle. The major features of Condor are automatic resource location and job allocation, check pointing and the migration of processes. These features are achieved without modification to the underlying UNIX kernel. However, it is necessary for a user to link their source code with Condor libraries. Condor monitors the activity on all the participating computing resources, those machines, which are determined to be available, are placed into a resource pool. Machines are then allocated from the pool for the execution of jobs. The pool is a dynamic entity – workstations enter when they become idle, and leave when they get busy.

- The Portable Batch System (PBS) [PBS] is a batch queuing and workload management system (originally developed for NASA). It operates on a variety of UNIX platforms, from clusters to supercomputers. The PBS job scheduler allows sites to establish their own scheduling policies for running jobs in both time and space. PBS is adaptable to a wide variety of administrative policies, and provides an extensible authentication and security model. PBS provides a GUI for job submission, tracking, and administrative purposes.

- The Sun Grid Engine (SGE) [SGE] is based on the software developed by Genias known as Codine/GRM. In the SGE, jobs wait in a holding area and queues located on servers provide the services for jobs. A user submits a job to the SGE, and declares a requirements profile for the job. As soon as a queue becomes available for execution of a new job, the SGE determines suitable jobs for the queue and will dispatch the job with the highest priority or longest waiting time; it will try to start new jobs in the least loaded and most suitable queue.

3.3.8 Storage Resource Broker

The Storage Resource Broker (SRB) [SRB] has been developed at San Diego Supercomputer Centre (SDSC) to provide "uniform access to distributed storage" across a range of storage devices, via a well-defined API. The SRB supports file replication, and this can occur either offline or on-the-fly. Interaction with the SRB is via a GUI. The SRB servers can be federated. The SRB is managed by an administrator, with authority to create user groups.

A key feature of the SRB is that it supports metadata associated with a distributed file system, such as location, size and creation date information. It also supports the notion of application-level (or domain-dependent) metadata, specific to the content and not generalisable across all data sets.

In contrast with traditional network file systems, SRB is attractive for grid applications in that it deals with large volumes of data, which can transcend individual storage devices, because it deals with metadata and takes advantage of file replication.

3.4 Grid portals on the Web

A web portal allows application scientists and researchers to access resources specific to a particular domain of interest via a web interface. Unlike typical web subject portals, a grid portal may also provide access to grid resources. For example a grid portal may authenticate users, permit them to access remote resources, help them make decisions about scheduling jobs, allowing users to access and manipulate resource information obtained and stored on a remote database. Grid portal access can also be personalised by the use of profiles which are created and stored for each portal user. These attributes, and others, make grid portals the appropriate means for e-Science users to access grid resources.

3.4.1 The NPACI HotPage

The NPACI HotPage [Hotpage] is a user portal that has been designed to be a single point-of-access to computer-based resources. It attempts to simplify access to resources that are distributed across member organisations and allows them to be viewed as either an integrated grid system or as individual machines.

The two key services provided by the HotPage include information and resource access and management services. The information services are designed to increase the effectiveness of users. It provides links to:

- User documentation and navigation;
- News items of current interest;
- Training and consulting information;
- Data on platforms and software applications;
- Information resources, such as user allocations and accounts.

Another key service offered by HotPage is that it provides status of resources and supports an easy mechanism for submitting jobs to resources. The status information includes:

- CPU load/percent usage;
- Processor node maps;
- Queue usage summaries;
- Current queue information for all participating platforms.

HotPage’s interactive web-based service offers secure transactions for accessing resources and allows the user to perform tasks such as command execution, compilation, and running programs.

3.4.2 The SDSC Grid Port Toolkit

The SDSC grid port toolkit [GridPort] is a reusable portal toolkit that uses HotPage infrastructure. The two key components of GridPort are the web portal services and the application APIs. The web portal module runs on a web server and provides a secure (authenticated) connectivity to the Grid. The application APIs provide a web interface that helps end-users develop customised portals (without having to know the underlying portal infrastructure). GridPort is designed to allow the execution of portal services and the client applications on separate web servers. The GridPortal toolkit modules have been used to develop science portals for applications areas such as pharmacokinetic modeling, molecular modelling, cardiac physiology, and tomography.

3.4.3 Grid Portal Development Kit

The Grid Portal Collaboration is an alliance between NCSA, SDSC and NASA IPG [NLANR]. The purpose of the Collaboration is to support a common set of components and utilities to make portal development easier and allow various portals to interoperate by using the same core infrastructure (namely the GSI and Globus).

Example portal capabilities include the following:

- Running simulations either interactively or submitted to a batch queue.
- File transfer including: file upload, file download, and third party file transfers (migrating files between various storage systems).
- Querying databases for resource/job specific information.

- Maintaining user profiles that contains information about past jobs submitted, resources used, results information and user preferences.

The portal architecture is based on a three-tier model, where a client browser securely communicates to a web server over a secure sockets (via `https`) connection. The web server is capable of accessing various grid services using the Globus infrastructure. The Globus toolkit provides mechanisms for securely submitting jobs to a Globus gatekeeper, querying for hardware/software information using LDAP, and a secure PKI infrastructure using GSI.

The portals discussion in this sub-section highlights the characteristics and capabilities that are required in grid environments. Portals are also relevant at the information and knowledge layers and the associated technologies are discussed in sections 4 and 5.

3.5 Grid Deployments

Having reviewed some of the most significant projects that are developing grid technologies, in this section we now describe some of the major projects that are deploying grid technologies. In particular we identify their aims and scope, to indicate the direction of current activities.

3.5.1 Information Power Grid

The main objective of NASA's Information Power Grid (IPG) [IPG] is to provide NASA's scientific and engineering communities with a substantial increase in their ability to solve problems that depend on the use of large-scale and/or distributed resources. The project team is focused on creating an infrastructure and services to locate, combine, integrate, and manage resources from across NASA centres, based on the Globus Toolkit. An important goal is to produce a common view of these resources, and at the same time provide for distributed management and local control. It is useful to note the development goals:

- Independent but consistent tools and services that support a range of programming environments used to build applications in widely distributed systems.
- Tools, services, and infrastructure for managing and aggregating dynamic collections of resources: processors, data storage/information systems, communications systems, real-time data sources and instruments, as well as human collaborators.
- Facilities for constructing collaborative, application-oriented workbenches and problem solving environments across NASA, based on the IPG infrastructure and applications.
- A common resource management approach that addresses areas such as systems management, user identification, resource allocations, accounting, and security.
- An operational grid environment that incorporates major computing and data resources at multiple NASA sites in order to provide an infrastructure capable of routinely addressing larger scale, more diverse, and more transient problems than is currently possible.

3.5.2 Particle Physics Grids

The EU Data Grid [DataGrid] has three principal goals: middleware for managing the grid infrastructure, a large scale testbed involving the upcoming Large Hadron Collider (LHC) facility, and demonstrators (production quality for high energy physics, also earth observation and biology). The DataGrid middleware is Globus-based.

The GriPhyN (Grid Physics Network) [GriPhyN] collaboration, led by the universities of Florida and Chicago, is composed of a team of experimental physicists and information technology researchers who plan to implement the first Petabyte-scale computational environments for data intensive science. The requirements arise from four physics experiments involved in the project. GriPhyN will deploy computational environments called Petascale Virtual Data Grids (PVDGs) that meet the data-intensive computational needs of a diverse community of thousands of scientists spread across the globe.

The Particle Physics Data Grid [PPG] is a consortium effort to deliver an infrastructure for very widely distributed analysis of particle physics data at multi-petabyte scales by hundreds to thousands of physicists. It also aims to accelerate the development of network and middleware infrastructure for data-intensive collaborative science.

3.5.3 EuroGrid and UNICORE

The EuroGrid [EuroGrid] is a project funded by the European Commission. It aims to demonstrate the use of grids in selected scientific and industrial communities, address the specific requirements of these communities, and highlight their use in the areas of biology, meteorology and computer-aided engineering.

The objectives of the EuroGrid project include the support of the EuroGrid software infrastructure, the development of software components, and demonstrations of distributed simulation codes from different application areas (biomolecular simulations, weather prediction, coupled CAE simulations, structural analysis, real-time data processing).

The EuroGrid software is UNICORE (UNiform Interface to COmputing REsources) [UNICORE] which has been developed for the German supercomputer centres. It is based on Java-2 and uses Java objects for communication, with the UNICORE Protocol Layer (UPL) handling authentication, SSL communication and transfer of data; Unicore pays particular attention to security.

3.6 Research Issues

The computation layer is probably the layer with the most software technology that is currently available and directly useable. Nevertheless, as this section has highlighted, distributed applications still offer many conceptual and technical challenges. The requirements in section 3.2 express some of these, and we highlight the following as key areas for further work:

1. Resource discovery – given a resource's unique name or characteristics there need to be mechanisms to locate the resource within the distributed system. Services are resources. Some resources may persist, some may be transitory, some may be created on demand.
2. Synchronisation and coordination – how to orchestrate a complex sequence of computations over a variety of resources, given the inherent properties of both loosely- and tightly-coupled distributed systems. This may involve process description, and require an event-based infrastructure. It involves scheduling at various levels, including metascheduling and workflow.
3. Fault tolerance and dependability – environments need to cope with the failure of software and hardware components, as well as access issues – in general, accommodating the exception-handling that is necessary in such a dynamic, multi-user, multi-organisation system.
4. Security – authentication, authorisation, assurance and accounting mechanisms need to be set in place, and these need to function in the context of increasing scale and automation. For example, a user may delegate privileges to processes acting on their behalf, which may in turn need to propagate some privileges further.
5. Concurrency and consistency – the need to maintain an appropriate level of data consistency in the concurrent, heterogeneous environment. Weaker consistency may be sufficient for some applications.
6. Performance – the need to be able to cope with non-local access to resources, through caching and duplication. Moving the code (or service) to the data (perhaps with scripts or mobile agents) is attractive and brings a set of challenges.
7. Heterogeneity – the need to work with a multitude of hardware, software and information resources, and to do so across multiple organisations with different administrative structures.
8. Scalability – systems need to be able to scale up the number and size of services and applications, without scaling up the need for manual intervention. This requires automation, and ideally self-organisation.

Although we believe the service-oriented approach will prove to be most useful at the information and knowledge layers, it is interesting to note that it can be applied to the various systems discussed in this section. In particular, it is compatible with the CORBA model, with the way Java is used as a distributed environment, and with Globus. We expect to see this trend continue in most of the significant grid technologies. We note that layered and service-based architectures are common in descriptions of the systems discussed in this section, notably Globus and IPG. For example, IPG has a 'grid common services layer'. Determining the core grid services in a service-oriented architecture is also a research issue.

Java and CORBA address several of the above issues and provide a higher level conceptual model, in both cases based on distributed object systems. However most currently deployed systems use Globus, with the Java-based UNICORE system an interesting alternative. A key question in all of this work, however, is what happens as scale increases? None of the systems considered in this section have demonstrated very large scale deployments, though some are set to do so. Pragmatically, some solutions will scale reasonably well, and the Web has demonstrated that scalability can be achieved; e.g. consider search engines. However we have significant concern about the need to cross organisational boundaries and the obstacles this will impose, relating to system management and security.

Many of these research issues are the subjects of working groups in the Global Grid Forum [GGF].

4. The Information Layer

In this section we focus firstly on the Web. The Web's information handling capabilities are clearly an important component of the e-Science infrastructure, and the web infrastructure is itself of interest as an example of a distributed system that has achieved global deployment. The second aspect addressed in this section is support for collaboration, something which is key to e-Science. We show that the web infrastructure itself lacks support for synchronous collaboration between users, and we discuss technologies that do provide such support.

It is interesting to consider the rapid uptake of the Web and how this might inform the design of the e-Science infrastructure, which has similar aspirations in terms of scale and deployment. One principle is clearly simplicity – there was little new in HTTP and HTML, and this facilitated massive deployment. We should however be aware of a dramatic contrast between Web and Grid: despite the large scale of the Internet, the number of hosts involved in a typical web transaction is still small, significantly lower than that envisaged for many grid applications.

The information layer aspects build on the idea of a 'collaboratory', defined in a 1993 US NSF study [Cerf93] as a "centre without walls, in which the nation's researchers can perform their research without regard to geographical location - interacting with colleagues, accessing instrumentation, sharing data and computational resource, and accessing information in digital libraries." This view accommodates 'information appliances' in the laboratory setting, which might, for example, include electronic logbooks and other portable devices.

The next section discusses technologies for the information layer and this is followed in 4.3 by consideration of support for collaboration. Section 4.4 considers the information layer aspects of the scenario.

4.1 Technologies for the Information Layer

4.1.1 The Web for Information Distribution

The early web architecture involved HTTP servers and web browsers, the browser simply being a client that supports multiple Internet protocols (including HTTP) and the then-new document markup language HTML. In responding to an HTTP request, the server can deliver a file or, using the Common Gateway Interface (CGI), invoke a local program (typically a script) that obtains or generates the content to be returned. The document returned by the server to the client is MIME-encoded so that the client can recognise its type. Different content types are handled at the browser by native support, invocation of a helper application or, more recently, browser 'plugins'. With forms, the client effectively sends a document rather than a simple request (in fact HTTP has always supported document uploads, but this is only now becoming used as authoring tools become better integrated).

Scalability is achieved through caching, whereby a copy of a document is stored at an intermediary between client and server so that future requests can be serviced without retrieving the entire document from the server. Familiar examples are the client-side caches maintained by web browsers, and ‘caching proxies’ shared by a group of users. Hierarchical caching schemes have also been explored. These involve a tree of parent and child caches in which each parent acts as a cache for the child caches below it. A cache consults all its neighbours if it does not have a document, and if none of them has it then it is obtained via a parent. Here there is a trade-off between performance and freshness of the content – there are fewer gains from accessing highly dynamic content, and some content is not cached at all. A server may specify an expiry time after which a document is out-of-date, after which the proxy server will check for a newer document. Documents that have changed recently are considered more likely to change again. Sometimes it is necessary to access a server directly, not via a cache, for reasons of access control (e.g. subscription to content).

The simplicity of the web architecture is constraining, and several techniques have emerged to overcome this. For example, forms and cookies are used to maintain state across multiple HTTP requests in order to have a concept of a session. This is achieved by the server returning state information to the client that is then sent back to the server in later requests. However, there are many practical advantages to the use of the strict client-server architecture and bi-directional synchronous communication over a single TCP/IP session. For example, once the client has connected to the server then the return path is immediately available; to establish the connection in the reverse direction may be prohibited by a firewall. There are performance penalties for establishing a new TCP/IP connection for every request, because it takes time for TCP/IP to establish a connection and for it to self-adjust to optimise performance to prevailing network conditions (‘slow start’). This has led to the introduction of ‘persistent connections’ in HTTP version 1.1 so that a connection can be reused over a close succession of requests. This is more efficient that using short connections in parallel.

For security, HTTP can be transported through the ‘secure socket layer’ (SSL) which uses encryption – this is ‘HTTPS’, and the related X509 standard deals with public key encryption. The evolution of SSL within IETF is called TLS (Transport Layer Security). Although HTTP is thought of as a reliable protocol, things can still go wrong and IBM have proposed reliable HTTP (HTTPR), a protocol for reliable messaging over HTTP.

4.1.2 Expressing Content and Metacontent

In section 3.1.2 we discussed the Web as an infrastructure for distributed applications, where information is exchanged between programs rather than being presented for a human reader. Such information exchange is facilitated by the XML family or recommendations from W3C.

XML is designed to mark up documents and has no fixed tag vocabulary; the tags are defined for each application using a Document Type Definition (DTD) or an XML Schema. A well-formed XML document is a labelled tree. Note that the DTD or Schema addresses syntactic conventions and does not address semantics. XML Schema are themselves valid XML expressions. Many new ‘formats’ are expressed in XML, such as SMIL (the synchronised multimedia integration language).

RDF (Resource Description Framework) is a standard way of expressing metadata, specifically resources on the Web, though in fact it can be used to represent structured data in general. It is based on ‘triples’ where each triple expresses the fact that an object O has attribute A with value V, written A(O,V). An object can also be a value, enabling triples to be ‘chained’, and in fact any RDF statement can itself be an object or attribute – this is called reification and permits nesting. RDF Schema are to RDF what XML Schema are to XML: they permit definition of a vocabulary. Essentially RDF schema provide a basic type system for RDF such as Class, subClassOf and subPropertyOf. RDF Schema are themselves valid RDF expressions.

XML and RDF (with XML and RDF schema) enable the standard expression of content and metacontent. Additionally a set of tools has emerged to work with these formats, for example parsers, and there is increasing support by other tools. Together this provides the infrastructure for the information layer. Other representational formats include the Ontology Interchange Language (OIL) and the DARPA Agent Markup Language (DAML), which have been brought together to form DAML+OIL. These are discussed in Section 5. W3C has created a Web Ontology Working Group to focus on the development of a language to extend the semantic reach of current XML and RDF metadata efforts.

4.1.3 Semantic Web

W3C ran a “Metadata Activity”, which addressed technologies including RDF, and this has been succeeded by the Semantic Web Activity. The activity statement [Semweb] describes the Semantic Web as follows:

> “The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications. The Web can reach its full potential if it becomes a place where data can be shared and processed by automated tools as well as by people.”

This vision is familiar – it shares much with the e-Science vision. The Scientific American paper [BernersLee01] provides motivation, with a scenario that uses agents. In a nutshell, the Semantic Web is intended to do for knowledge representation what the Web did for hypertext. The Semantic Web activity now includes The Web Ontology Working Group, which will build upon the RDF core work. Section 5 focuses on Semantic Web and associated technologies.

4.1.4 Towards an Adaptive Information Grid

Having focused on process-to-process information exchange, in this section we revisit the interface to the human. A key goal of the Semantic Grid is to provide the e-Scientist with the right information at the right time, i.e. personalisation and a degree of context-awareness. This requirement is amplified by the huge scale of information that will be generated by e-Science.

Content can be generated automatically or pre-existing content can be transformed dynamically according to circumstances. To this end, the sets of links (hyperstructure) support navigation around a body of documents and adapting these links is a powerful mechanism for personalisation which might not require dynamic content generation (rather it provides a different view on a set of pre-existing information resources, which can be a useful separation of concerns). The lifetime of dynamically generated content must be considered – for example, it may need to persist so that it can be annotated.

Although this is a well-established approach within hypermedia systems, the web linking technology is particularly problematic. The basic problem is that once a document is published, anybody on the planet can embed a link to it within their own documents. That link is then hardwired, and it is also vulnerable to decay as the original document might be renamed or moved. An alternative architecture (known as 'open' hypermedia [Davis99]) maintains the link information separately from the document. This facilitates dynamic generation of the hyperstructures and maintenance of link integrity. Web standards have evolved to accommodate this mode of working, supporting 'out of line' links.

In contrast, current generation search engines are a good example of a service that is not adaptive – typically two different users with the same query will get the same result. Ideally the search query will be qualified by the user's context, and the results ranked accordingly on return. The ultimate search engine, however, would be one that accepted an empty query and gave the right results, entirely based on information about the user and what they are doing. Metasearch takes advantage of several search engines and typically involves some form of query expansion and query routing, a process that can also be adapted dynamically.

To illustrate the creation of user models, consider the following two techniques. Firstly, the user might specify their interests, perhaps on a form or by providing a representative set of documents (perhaps from bookmarks). Secondly, the model might be obtained by incidental capture of information based on browsing habits. The latter can occur close to the server (e.g. from server logs), near the client (where detailed information, such as dwell time and other actions on the document can be recorded) or at an intermediate stage. The set of documents visited by a user is their *trail*, and based on trails it is possible to help users find other users who have travelled (or are travelling) the same route.

A popular mechanism to realise adaptive systems in the web context is the use of proxies, using them to perform other functions as the requests pass through and documents pass back [Barrett98]. There are many examples:

- Caching proxies, including hierarchical caching in which caches are arranged as parents and siblings.
- Proxies producing logs. This enables logging local to part of a network, rather than using server logs. Logs can be used for management purposes but also to capture user browsing behaviour as part of an adaptive information system.
- Proxies modifying document content. Simple modifications include rewriting URLs, introducing links, prepending or appending content. With a full parser, the proxy can do more sophisticated transformations (but may take more time to do it).
- Proxies modifying requests. The proxy can rewrite a request, or respond to a request by redirecting the client.

Another technique to extend functionality is to download code to the client, e.g. as Java or JavaScript. Such applications are constrained by the execution environment and typically only provide additional interactive functionality. Note that a downloaded Java application (applet) is able to initiate a connection to the server, enabling new bi-directional communication channels to be opened up. For example, this enables a server to send live data to the client without the client polling for it.

What we have described in this section is current web practise which is relevant to the information grid. Building on top of this, however, we envisage an "adaptive grid" being developed, and this will be further supported by the emerging technologies described in section 5.2.

The role of the browser is then to provide the human interface. Web browsers have become the standard user interface to many services and applications, where they provide a widely available and highly configurable interface. However in the fullness of time it is clear that conventional browsers will not be the only form of interface to web-based information systems; for example, with augmented reality, queries will be made via many forms of device, and responses will appear via others.

4.1.5 The Web as an e-Science Information Infrastructure

The Web was originally created for distribution of information in an e-Science context at CERN. So an obvious question to ask is does this information distribution architecture described in 4.1.1-3 meet grid requirements? A number of concerns arise:

- Version control. The popular publishing paradigm of the Web involves continually updating pages without version control. In itself the web infrastructure does not explicitly support versioning.
- Quality of service. Links are embedded, hardwired global references and they are fragile, rendered useless by changing the server, location, name or content of the destination document. Expectations of link consistency are low and e-Science may demand a higher quality of service.
- Provenance. There is no standard mechanism to provide legally significant evidence that a document has been published on the Web at a particular time [Probity][Haber91].
- Digital Rights Management. e-Science demands particular functionality with respect to management of the digital content, including for example copy protection and intellectual property management.
- Curation. Much of the web infrastructure focuses on the machinery for delivery of information rather than the creation and management of content. Grid infrastructure designers need to address metadata support from the outset (this issue is more fully justified in section 5).

To address some of these issues we can look to work in other communities. For example, the multimedia industry also demands support for digital rights management. MPEG-21 aims to define 'a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities' [MPEG21], addressing the multimedia content delivery chain. It is interesting to consider its seven elements in the context of grid computing:

1. Digital Item Declaration - schema for declaring digital items
2. Digital Item Identification and Description - framework for identification and description of any entity
3. Content Handling and Usage - interfaces and protocols for creation, manipulation, search, access, storage, delivery and content use and reuse
4. Intellectual Property Management and Protection
5. Terminals and Networks - transparent access to content across networks and terminals
6. Content Representation
7. Event Reporting – for users to understand the performance of all reportable events within the framework.

Authoring is another major concern, especially collaborative authoring. The Web-based Distributed Authoring and Versioning (WebDAV) activity [WebDAV] is chartered "to define the HTTP extensions necessary to enable distributed web authoring tools to be broadly interoperable, while supporting user needs".

In summary, although the Web provides an effective layer for information transport, it does not provide a comprehensive information infrastructure for e-Science.

4.1.6 Information Requirements of the Infrastructure

We can view the e-Science infrastructure as a number of interacting components, and the information that is conveyed in these interactions falls into a number of categories. One of those is the domain specific content that is being processed. Additional types include:

- Information about components and their functionalities within the domain
- Information about communication with the components
- Information about the overall workflow and individual flows within it

These must be tied down in a standard way to promote interoperability between components, with agreed common vocabularies. By way of example, Agent Communication Languages (ACLs) address exactly these issues. In particular the Foundation for Intelligent Physical Agents (FIPA) activity [FIPA], and more recently DAML-S, provide approaches to establishing a semantics for this information in an interoperable manner. FIPA produces software standards for heterogeneous and interacting agents and agent-based systems, including extensive specifications. In the FIPA abstract architecture:

- Agents communicate by exchanging messages which represent speech acts, and which are encoded in an agent-communication-language.
- Services provide support agents, including directory-services and message-transport-services.
- Services may be implemented either as agents or as software that is accessed via method invocation, using programming interfaces (e.g. in Java, C++ or IDL).

Again we can identify agent-to-agent information exchange and directory entries as information formats which are required by the infrastructure 'machinery'.

4.1.7 Web Services

The two recent developments in the web community that have attracted most interest are Web Services and the Semantic Web. Web Services, outlined in this section, provide a potential realisation of certain aspects of the service-oriented architecture proposed in section 2. The Semantic Web, on the other hand, utilises the RDF and RDF Schema technologies and is discussed in section 5.

We have seen that the web architecture for document delivery is different to the technologies used for distributed applications, and that the web infrastructure is increasingly supportive of process-to-process information exchange. Web Services represent the convergence of these technologies. The Web Services proposals are industry led, and are in various stages of the W3C process:

- *XML Protocol*. Originally this was the *Simple Object Access Protocol* (SOAP). Essentially, XML Protocol [XMLP] enables applications to access remote applications; it is often characterised as an XML-based remote procedure call mechanism. XML Protocol can be used for exchanging any XML information. SOAP is a working draft from W3C, together with XML Protocol Abstract Model (the set of concepts that XML Protocol must adhere to).
- *Web Services Description Language* (WSDL). Describes the service (in a limited way) and how to use it, in some ways similar to an IDL. WSDL is available as a W3C note [WSDL].
- *Universal Description Discovery and Integration* (UDDI). This is a directory of registered services, similar to a 'yellow pages'. UDDI support 'publish, find and bind': a service provider publishes the service details to the directory; service requestors make requests to the registry to find the providers of a service; the services 'bind' using the technical details provided by UDDI. Effectively, UDDI integrates yellow and white page services because it has both business services; it additionally provides the 'binding templates' [UDDI].

The next service attracting interest is at the process level. For example, Web Services Flow Language (WSFL) [WSFL] is an IBM proposal that defines workflows as combinations of Web Services and enables workflows themselves to appear as services; XLANG [XLANG] from Microsoft supports complex transactions that may involve multiple Web Services. It may also be useful to be able to decompose services, replacing elements to meet individual needs

4.2 Live Information Systems

4.2.1 Collaboration

As an information dissemination mechanism the Web might have involved many users as 'sinks' of information published from major servers. However in practice, part of the web phenomenon has been widespread publishing by the users. This has had a powerful effect in creating online communities. However, the paradigm of interaction is essentially ‘publishing things at each other’, and is reinforced by email and newsgroups which also support asynchronous collaboration.

Despite this, however, the underlying internet infrastructure is entirely capable of supporting live (real-time) information services and synchronous collaboration. For example:

- Live data from experimental equipment
- Live video feeds (‘webcams’) via unicast or multicast (e.g. MBONE).
- Videoconferencing (e.g. H.323, coupled with T.120 to applications, SIP)
- Internet Relay Chat.
- MUDs
- Chat rooms
- Collaborative Virtual Environments

All of these have a role in supporting e-Science, directly supporting people, behind the scenes between processes in the infrastructure, or both. In particular they support the extension of e-Science to new communities that transcend current organisational and geographical boundaries.

Although the histories of these technologies predate the Web, they can interoperate with the Web and build on the web infrastructure technologies through adoption of appropriate standards. For example, messages can be expressed in XML and URLs are routinely exchanged. In particular the web’s metadata infrastructure has a role: data from experimental equipment can be expressed according to an ontology, enabling it to be processed by programs in the same way as static data such as library catalogues.

The application of computer systems to augment the capability of humans working in groups has a long history, with origins in the work of Doug Englebart [Englebart62]. In this context, however, the emphasis is on facilitating distributed collaboration, and we wish to embrace the increasingly ‘smart’ workplaces of the e-Scientist including meeting rooms and laboratories. Amongst the considerable volume of work in the ‘smart space’ area we note in particular the Smart Rooms work by Pentland [Pentland96] and Coen’s work on the Intelligent Room [Coen98]. This research area falls under the “Advanced Collaborative Environments” Working group of the Global Grid Forum (ACE Grid), which addresses both collaboration environments and ubiquitous computing.

4.2.2 Access Grid

The Access Grid is a multicast videoconferencing infrastructure to support the collaboration of e-Scientists. In the UK there will be access grid nodes at the national and eight regional e-Science centres [AccessGrid]. Multicast videoconferencing is a familiar infrastructure in the UK in the form of the JANET multicast backbone (MBONE) which has been in service since 1991, first using tunnels and as a native service since 2000; international native multicast peerings have also been established.

The ISDN-based videoconferencing world (based on H.320) has evolved alongside this, and the shift now is to products supporting LAN-based videoconferencing (H.323). In this world, the T.120 protocol is used for multicast data transfer, such as remote camera control and application sharing. Meanwhile the IETF has developed Session Initiation Protocol (SIP), which is a signalling protocol for establishing real-time calls and conferences over internet networks. This resembles HTTP and uses Session Description Protocol (SDP) for media description.

During a meeting there is live exchange of information, and this brings the information layer aspects to the fore. For example, events in one space can be communicated to other spaces to facilitate the meeting. At the simplest level this might be slide transitions or remote camera control. These provide metadata which is generated automatically by software and devices, and can be used to enrich the conference and stored for later use. New forms of information may need to be exchanged to handle the large scale of meetings, such as distributed polling and voting.

Another source of live information is the notes taken by members of the meeting, or the annotations that they make on existing documents. Again these can be shared and stored to enrich the meeting. A feature of current collaboration technologies is that sub-discussions can be created easily and without intruding – these also provide enriched content.

In videoconferences, the live video and audio feeds provide presence for remote participants – especially in the typical access grid installation with three displays each with multiple views. It is also possible for remote participants to establish other forms of presence, such as the use of avatars in a collaborative virtual environment. For example, participants can share a 3D visualisation of the meeting spaces. This convergence of the digital and physical – where people are immersed in a virtual meeting space and/or remote participants are 'ghosts' in the physical meeting space – is the area of the Equator project, one of the Interdisciplinary Research Collaborations funded by the EPSRC in 2000 [EQUATOR].

The combination of Semantic Web technologies with live information flows is highly relevant to grid computing and is an emerging area of activity [Page01]. Metadata streams may be generated by people, by equipment or by programs – e.g. annotation, device settings, data processed in real-time. Live metadata in combination with multimedia streams (such as multicast video) raises quality of service (QoS) demands on the network and raises questions about whether the metadata should be embedded (in which respect, the multimedia metadata standards are relevant).

4.3 Information Layer Aspects of the Scenario

To realise the scenario, and the information services in table 2.2, the information layer of the Semantic needs to deal with a variety of information types. These are identified in the table 4.1, with comments on the content representation and metadata.

Information	**Representation**	**Description (metadata)**
Sample ID	RDF	
Analysis results	Raw or XML	ID, timestamp, parameters
Analyser configurations used in previous runs	XML	parameters
Video	MPEG etc	ID, timestamp (RDF), events
A URL		
Agenda	XML	Author etc
Videoconference	RTP etc	Participants, slide transitions
Published analysis results	XML	RDF catalogue data
Notifications of new results	XML or RDF	
Community publications	raw	RDF bibliographic data
Service descriptions	WSDL	

Table 4.1: Information in the scenario

Each of these pieces of information requires a common understanding by those parties using it at the time and many require an understanding for retrospective use. There is an additional form of information, key to provenance and to automation: the description of the workflow. This could be described in a language such as WSFL. The discovery of services may involve a registry (such as UDDI) which does not appear explicitly in the scenario. There will also be security and provenance information (e.g. certificates, digests), as well as cost information for charging and other "housekeeping" information. Exception handling will also result in information flow and is an important area that may be too readily overlooked.

4.4 Research Issues

Although many of the technologies discussed in this section are available today (even if only in a limited form), a number of the topics still require further research. These include:

1. Issues relating to e-Science content types. Caching when new content is being produced. How will the web infrastructure respond to the different access patterns resulting from automated access to information sources? Issues in curation of e-Science content.
2. Digital rights management in the e-Science context (as compared with multimedia and e-commerce, for example).
3. Provenance. Is provenance stored to facilitate reuse of information, repeat of experiments, or to provide evidence that certain information existed at a certain time?
4. Creation and management of metadata, and provision of tools for metadata support.
5. Service descriptions, and tools for working with them. How best does one describe a service-based architecture?
6. Workflow description and enaction, and tools for working with descriptions.
7. Adaptation and personalisation. With the system 'metadata-enabled' throughout, how much knowledge can be acquired and how can it be used?

8. Collaboration infrastructure for the larger community, including interaction between scientists, with e-Science content and visualisations, and linking smart laboratories and other spaces.
9. Use of metadata in collaborative events, especially live metadata; establishing metadata schema to support collaboration in meetings and in laboratories.
10. Capture and presentation of information using new forms of device; e.g. for scientists working in the field.
11. Interplay between ‘always on’ devices in the e-Scientist’s environment and portable devices with local storage.
12. Repesentation of information about the underlying grid fabric, as required by applications; e.g. for resource scheduling and monitoring.

5. The Knowledge Layer

The aim of the knowledge layer is to act as an infrastructure to support the management and application of scientific knowledge to achieve particular types of goal and objective. In order to achieve this, it builds upon the services offered by the data-computation and information layers.

The first thing to reiterate at this layer is the problem of the sheer scale of content we are dealing with. We recognise that the amount of data that the data grid is managing will be huge. By the time that data is equipped with meaning and turned into information we can expect order of magnitude reductions in the amount. However the amount of information remaining will certainly be enough to present us with a problem – a problem recognised as *infosmog* – the condition of having too much information to be able to take effective action or apply it in an appropriate fashion to a specific problem. Once information is delivered that is destined for a particular purpose, we are in the realm of the knowledge grid that is fundamentally concerned with abstracted and annotated content, with the management of scientific knowledge.

We can see this process of scientific knowledge management in terms of a life cycle (figure 5.1) of knowledge-oriented activity that ranges over knowledge acquisition and modelling, knowledge retrieval and reuse, knowledge publishing and knowledge maintenance (section 5.1). In the rest of this section we first review the knowledge life cycle in more detail. Next we discuss the fundamental role that ontologies will play in providing semantics for the knowledge layer (section 5.2). Section 5.3 then reviews the current state of knowledge technologies – that is tools and methods for managing the sort of knowledge content that will be supported in the knowledge grid. Section 5.4 then considers how the knowledge layers of the Grid would support our extended scenario. Finally, we review the research issues that arise out of our requirements for a knowledge grid (section 5.5).

5.1 The Knowledge Lifecycle

Although we often suffer from a deluge of data and too much information, all too often what we have is still insufficient or too poorly specified to address our problems, goals and objectives. In short, we have insufficient knowledge. *Knowledge acquisition* sets the challenge of getting hold of the information that is around, and turning it into knowledge by making it *usable*. This might involve, for instance, making tacit knowledge explicit, identifying gaps in the knowledge already held, acquiring and integrating knowledge from multiple sources (e.g. different experts, or distributed sources on the web), or acquiring knowledge from unstructured media (e.g. natural language or diagrams).

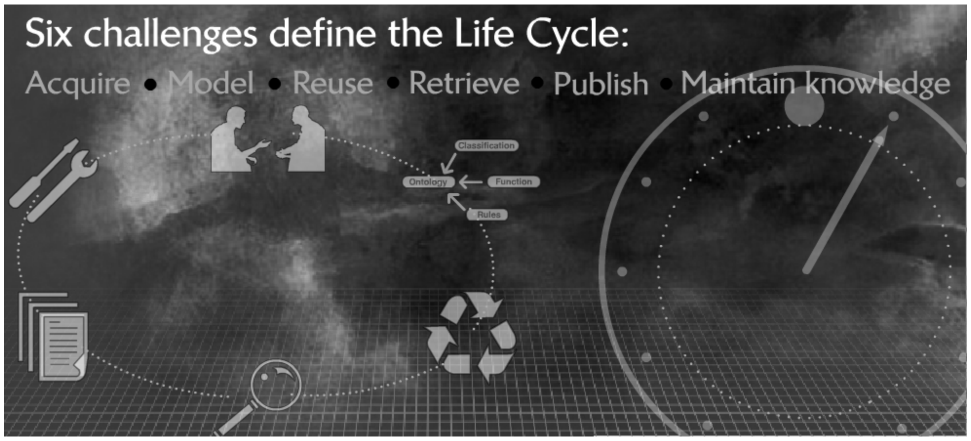
Six challenges define the Life Cycle:
Acquire • Model • Reuse • Retrieve • Publish • Maintain knowledge
Classification
Ontology
Function
Rules

Figure 5.1: The knowledge life cycle

Knowledge modelling bridges the gap between the acquisition of knowledge and its use. Knowledge models must be able *both* to act as straightforward placeholders for the acquired knowledge coming in, *and* to represent the knowledge so that it can be used for problem-solving.

Once knowledge has been acquired and modelled, one hopes it will be stored or hosted somewhere meaning that we need to be able to retrieve it efficiently. There are two related problems to do with *knowledge retrieval*. First, there is the issue of finding knowledge again once it has been stored. And second, there is the problem of retrieving the subset of content that is relevant to a particular problem. This second problem may well set problems for a knowledge retrieval system where that knowledge alters regularly and quickly during problem-solving.

One of the most serious impediments to the cost-effective use of knowledge is that too often knowledge components have to be constructed afresh. There is little *knowledge reuse*. This arises partly because knowledge tends to require different representations depending on the problem-solving that it is intended to do. We need to understand how to find patterns in knowledge, to allow for its storage so that it can be reused when circumstances permit. This would save a good deal of effort in reacquiring and restructuring the knowledge that had already been used in a different context.

Having acquired knowledge, modelled and stored it, the issue then arises as to how to get that knowledge to the people who subsequently need it. The challenge of *knowledge publishing* or disseminating can be described as getting the right knowledge, in the right form, to the right person or system, at the right time and is analogous to many of the issues raised in section 4.1.2 at the information layer. Different users and systems will require knowledge to be presented and visualised in different ways. The quality of such presentation is not merely a matter of preference. It may radically affect the utility of the knowledge. Getting presentation right will involve understanding the different perspectives of people with different agendas and systems with different requirements. An understanding of knowledge content will help to ensure that important related pieces of knowledge get published at the appropriate time.

Finally, having got the knowledge acquired, modelled and having managed to retrieve and disseminate it appropriately, the last challenge is to keep the knowledge content current – *knowledge maintenance*. This may involve the regular updating of content as content changes. But it may also involve a deeper analysis of the knowledge content. Some content has considerable longevity, while other knowledge dates very quickly. If knowledge is to remain active over a period of time, it is essential to know which parts of the knowledge base must be discarded and when. Other problems involved in maintenance include verifying and validating the content, and certifying its safety.

If the knowledge intensive activities described above are to be delivered in a grid context we will need to build upon and extend the main elements of the proposed Semantic Web. It is to this we now turn.

5.2 Ontologies and the Knowledge Layer

While the basic concepts and languages of the Semantic Web (as introduced in section 4) are generally appropriate for specifying and delivering services at the information layer, it is generally agreed that they lack the expressive power to be used as the basis for modelling and reasoning with knowledge[6]. To this end, the concept of an *ontology* is necessary. Generally speaking, an ontology determines the extension of terms and the relationships between them. However, in the context of knowledge and web engineering, an ontology is simply a published, more or less agreed, conceptualization of an area of content. The ontology may describe objects, processes, resources, capabilities or whatever.

Recently a number of languages have appeared that attempt to take concepts from the knowledge representation languages of AI and extend the expressive capability of RDF and RDF Schema. Examples include SHOE [Luke00], DAML [Hendler00], and OIL [vanHarmelen00]. Most recently there has been an attempt to integrate the best features of these languages in a hybrid called DAML+OIL. As well as incorporating constructs to help model ontologies DAML+OIL is being equipped with a logical language to express rule-based generalizations. The W3C Web Ontology Working Group, part of the Semantic Web Activity, is focusing on the development of a language to extend the semantic reach of current XML and RDF metadata efforts.

However the development of the Semantic Web is not simply about producing machine-readable languages to facilitate the interchange and integration of heterogeneous information. It is also about the elaboration, enrichment and annotation of that content. To this end, the list below is indicative of how rich annotation can become. Moreover it is important to recognize that enrichment or meta-tagging can be applied at any conceptual level in the three tier grid. This yields the idea of meta-data, meta-information and meta-knowledge.

- Domain ontologies: Conceptualisations of the important objects, properties and relations between those objects. Examples would include an agreed set of

[6] When building or modeling ontologies there are certain representational and semantic distinctions that it is important to be able to model. Examples would be the necessary and sufficient conditions for membership of a class and whether two classes are equivalent or disjoint. However RDF and RDF Schema lack these capabilities. Neither do they have the ability to model general constraints.

annotations for medical images, an agreed set of annotations for climate information, and a controlled set of vocabulary for describing significant features of engineering design.

- Task ontologies: Conceptualisations of tasks and processes, their interrelationships and properties. Examples would include an agreed set of descriptors for the stages of a synthetic chemistry process, an agreed protocol for describing the dependencies between optimisation methods, and a set of descriptions for characterizing the enrichment or annotation process when describing a complex medical image.

- Quality ontologies: Conceptualisations of the attributes that knowledge assets possess and their interrelationships. Examples would include annotations that would relate to the expected error rates in a piece of medical imaging, the extent to which the quality of a result from a field geologist depended on their experience and qualifications, and whether results from particular scientific instruments were likely to be superseded by more accurate devices.

- Value ontologies: Conceptualisations of those attributes that are relevant to establishing the value of content. Examples would include the cost of obtaining particular physics data, the scarcity of a piece of data from the fossil record, and how widely known a particular metabolic pathway was.

- Personalisation ontologies: Conceptualisations of features that are important to establishing a user model or perspective. Examples would include a description of the prior familiarity that a scientist had with particular information resources, the amount of detail that the user was interested in, and the extent to which the user's current e-Science activities might suggest other content of interest.

- Argumentation ontologies – A wide range of annotations can relate to the *reasons why* content was acquired, why it was modelled in the way it was, and who supports or dissents from it. This is particularly powerful when extended to the concept of associating discussion threads with content. Examples are the integration of authoring and reviewing processes in on-line documents. Such environments allow structured discussions of the evolution and development of an idea, paper or concept. The structured discussion is another annotation that can be held in perpetuity. This means that the reason for a position in a paper or a design choice is linked to the object of discussion itself.

The benefits of an ontology include improving communication between systems whether machines, users or organizations. They aim to establish an agreed and perhaps normative model. They endeavour to be consistent and unambiguous, and to integrate a range of perspectives. Another benefit that arises from adopting an ontology is inter-operability and this is why they figure large in the vision for the Semantic Web. An ontology can act as an interlingua, it can promote reuse of content, ensure a clear specification of what content or a service is about, and increase the chance that content and services can be successfully integrated.

A number of ontologies are emerging as a consequence of commercial imperatives where vertical marketplaces need to share common descriptions. Examples include the Common Business Library (CBL), Commerce XML (cXML), ecl@ss, the Open Applications Group Integration Specification (OAGIS), Open Catalog Format (OCF), the Open Financial Exchange (OFX), Real Estate Transaction Markup Language (RETML), RosettaNet, UN/SPSC (see www.diffuse.org), and UCEC. Moreover, there are a number of large-scale ontology initiatives underway in specific scientific communities. One such is in the area of genetics where a great deal of effort was invested in producing common terminology and definitions to allow scientists to manage their knowledge (www.geneontology.org/). This effort provides a glimpse of how ontologies will play a critical role in sustaining the e-Scientist.

This work can also be exploited to facilitate the sharing, reuse, composition, mapping, and succinct characterizations of (web) services. In this vein, [McIlraith01] exploit a web service markup that provides an agent-independent *declarative API* that is aimed at capturing the data and metadata associated with a service together with specifications of its properties and capabilities, the interface for its execution, and the prerequisites and consequences of its use. A key ingredient of this work is that the markup of web content exploits ontologies. They have used DAML for *semantic markup* of Web Services. This provides a means for agents to populate their local knowledge bases so that they can reason about Web Services to perform automatic web service discovery, execution, composition and interoperation.

More generally speaking, we can observe a variety of e-Business infrastructure companies who are beginning to announce platforms to support some level of Web Service automation. Examples of such products include Hewlett-Packard's e-speak, a description, registration, and dynamic discovery platform for e-services. Microsoft's .NET and BizTalk tools; Oracle's Dynamic Services Framework; IBM's Application Framework for e-Business; and Sun's Open Network Environment. The company VerticalNet Solutions is building ontologies and tools to organize and customize web service discovery. Its OSM Platform promises an infrastructure that is able to coordinate Web Services for public and private trading exchanges. These developments are very germane not only for e-Business but also for e-Science.

It can be seen that ontologies clearly provide a basis for the communication, integration and sharing of content. But they can also offer other benefits. An ontology can be used for improving search accuracy by removing ambiguities and spotting related terms, or by associating the information retrieved from a page with other information. They can act as the backbone for accessing information from a community web portal [Staab00]. Internet reasoning systems are beginning to emerge that exploit ontologies to extract and generate annotations from the existing web [Decker99].

Given the developments reviewed in this section, a general process that might drive the emergence of the knowledge grid would comprise:

- The development, construction and maintenance of application (specific and more general areas of science and engineering) and community (sets of collaborating scientists) based ontologies.

- The large scale annotation and enrichment of scientific data, information and knowledge in terms of these ontologies
- The exploitation of this enriched content by knowledge technologies.

5.3 Technologies for the Knowledge Layer

Given the processes outlined above, this section deals with the state of those technologies that might contribute to the construction and exploitation of annotated knowledge content, and to the general life cycle of knowledge content.

We have reviewed some of the language development work that is being undertaken to provide capabilities for expressive ontology modeling and content enrichment. The W3C has recently convened a Semantic Web activity (www.w3.org/2001/sw) that is looking into the options available and a community portal (www.semanticWeb.org) is in existence to give access to a range of resources and discussion forums. It is fair to say that at the moment most of the effort is in building XML (www.w3.org/XML, www.xml.com) and RDF (www.w3.org/RDF) resources that, whilst limited, do give us ways and means to build ontologies and annotate content.

Tools to build ontologies are thin on the ground but recently one of the best has become open source and is attracting a lot of external development work. Protégé 2000 is a graphical-based software tool developed at Stanford. Protégé is conceptually clear and supports both the import and export of ontologies in RDF, RDFS, and XML. The currently available versions of Protégé2000 do not provide annotation tools to map information from an ontology to related web content. Currently the process involves inserting the XML or RDF annotations into the content manually. There is clearly an urgent need for the semi-automatic annotation of content.

Besides ontology construction and annotation there are many other services and technologies that we need in our knowledge grid:

- services that support knowledge discovery methods that seek to locate patterns in information sets,
- services to cluster and index large amounts of content,
- services that will provide mappings between one set of ontologies and another,
- services that dynamically annotate content and link it up according to a particular conceptual scheme,
- services that précis large amount of content,
- services that provide customized visualizations of large content sets,
- services that perform substantial task oriented reasoning such as scheduling, monitoring, diagnosis and assessment.

The need for these services is leading to the emergence of controlled vocabularies for describing or advertising capabilities – one could almost say ontologies of service types and competencies. Some of these have already been reviewed in section 4 and include UDDI specification (www.uddi.org); ebXML (www.ebXML.org); and eSpeak (www.e-speak.hp.com).

A number of these services are going to require inference engines capable of running on content distributed on the knowledge grid. Inference has been an important component in the visions of the Semantic Web that have been presented to date [BernersLee99,01]. According to these views, agents and inference services will gather knowledge expressed in RDF from many sources and process this to fulfil some task. However most Semantic Web-enabled inference engines are little more than proofs of concept. SiLRI [Decker98] for example is an F-Logic inference engine that has the advantage of greater maturity over the majority of other solutions. In the US, work on the Simple HTML Ontology Extensions (SHOE) has used for example Parka, a high-performance frame system [Stoffel97]; and XSB, a deductive database [Sagonas94]; to reason over annotations crawled out of Semantic Web content. There are other inference engines that have been engineered to operate on the content associated with or extracted from web pages. Description Logics are particularly well suited to inferences associated with ontological structures – for example inferences associated with inheritance, establishing which classes particular instances belong to and providing the means to reorganize ontologies to capture generalities whilst maintaining maximum parsimony [Horrocks99], see also www.cs.man.ac.uk/~horrocks/FaCT .

However, with all of these inference engines there are likely to be problems of scale and consistency depending on the number and quality of annotations crawled out of enriched content. How can we ensure that any of our automated inference methods deliver results in which we can trust? Trust spans a range of considerations but includes at least the following important considerations:

- Are the facts interred in the web annotations correct or complete?
- Is the inference method sound?
- Is the inference method considering all the relevant content?

A different perspective on undertaking reasoning on the Grid or Web is to move away from generic methods applicable to any annotated content and concentrate instead on task specific reasoning. The emergence of problem-solving environments (PSEs) takes this position. These systems allow the user to exploit resources to solve particular problems without having to worry about the complexities of grid fabric management. As Gannon and Grimshaw [Gannon99] note "these systems …allow users to approach a problem in terms of the application area semantics for which the PSE was designed". Examples include the composition of suites of algorithms to solve for example design optimization tasks. Consider the design optimisation of a typical aero-engine or wing (see figure 5.2). It is necessary (1) to specify the wing geometry in a parametric form which specifies the permitted operations and constraints for the optimisation process, (2) to generate a mesh for the problem (though this may be provided by the analysis code), (3) decide which code to use for the analysis, (4) decide the optimisation schedule, and finally (5) execute the optimisation run coupled to the analysis code.

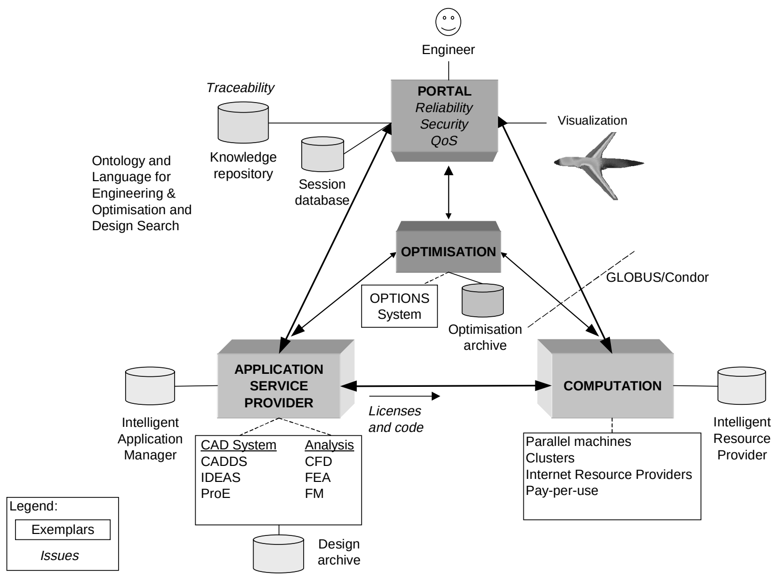

Figure 5.2: A web service knowledge enabled PSE [Geodise]

In the type of architecture outlined above, each of the components is viewed as a web service. It is therefore necessary to wrap each component using, for example, open W3C standards by providing an XML schema and using XML Protocol to interact with it – in other words an ontology. However, often the knowledge in a human designer's mind as to how to combine and set up a suite of tasks to suit a particular domain problem remains implicit. One of the research issues confronting architectures such as that outlined above is to start to make the designer's procedural knowledge explicit and encode it within PSEs.

A similar approach to specialist problem solving environments on the Web originates out of the knowledge engineering community. In the IBROW project (http://www.swi.psy.uva.nl/projects/ibrow/home.html) the aim is the construction of Internet reasoning services. These aim to provide a semi-automated facility that assists users in selecting problem-solving components from online libraries and configuring them into a running system adapted to their domain and task. The approach requires the formal description of the capabilities and requirements of problem-solving components. Initial experiments have demonstrated the approach for classification problem solving.

Providing complete reasoning services will remain a difficult challenge to meet. However, there are a variety of technologies available and under research (www.aktors.org) to support the more general process of knowledge acquisition, reuse, retrieval, publishing and maintenance.

Capturing knowledge and modelling it in computer systems has been the goal of knowledge-based systems (KBS) research for some 25 years [Hoffman95]. For example, commercial tools are available to facilitate and support the elicitation of knowledge from human experts [Milton99]. One such technique, the repertory grid, helps the human expert make tacit knowledge explicit.

KBS research has also produced methodologies to guide the developer through the process of specifying the knowledge models that need to be built. One such methodology, CommonKADS [Schrieber00], guides the process of building and documenting knowledge models. Since, the development of knowledge intensive systems is a costly and lengthy process it is important that we are able to re-use knowledge content. CommonKADS is based around the idea of building libraries of problem solving elements and domain descriptions that can be reused.

Knowledge publishing and dissemination is supported through a range of document management systems that provide more or less comprehensive publication services. Many are now exploiting content mark up languages to facilitate the indexing, retrieval and presentation of content. As yet few of them are powerfully exploiting the customization or personalization of content. However, within the UK EPSRC funded Advanced Knowledge Technologies (www.aktors.org) or AKT project, there is a demonstration of how content might be personalized and delivered in a way determined by a user's interests expressed via an ontology (http://eldora.open.ac.uk/my-planet/).

One of the interesting developments in knowledge publishing is the emergence of effective peer-to-peer publication archiving. The ePrints initiative (www.eprints.org) provides a range of services to allow publications to be archived and advertised with extended meta-data that will potentially allow a variety of knowledge services to be developed. For example, content-based retrieval methods. Workflow tools also exist that help support organization procedures and routines. For example, tools to support the collection, annotation and cataloguing of genomic data. However, they often use proprietary standards.

One of the key capabilities is support for communication and collaborative work and this was discussed in section 4.2.1. A range of web conferencing tools and virtual shared workspace applications is increasingly facilitating dialogue and supporting communication between individuals and groups. Netmeeting using MCU technology to support multicasting over the JANET is being trailed by UKERNA. CVW developed by Mitre corporation presents a rich environment for virtual collaborative meeting spaces. Digital whiteboards and other applications are also able to support brainstorming activities between sites. More ambitious visions of collaboration can be found in teleimmersive collaborative design proposals [Gannon99] exploiting CAVE environments. This aspect of large-scale immersive technology and the mix of real and virtual environments is at the heart of the research agenda for EQUATOR IRC (http://www.equator.ac.uk).

As we noted in section 4.1.2, one of the core components to be inserted into the UK e-Science Regional Centres are Access Grids. Access Grids support large-scale distributed meetings, collaborative teamwork sessions, seminars, lectures, tutorials, and training. An Access Grid node consists of large-format multimedia display, presentation, and interaction software environments. It has interfaces to grid middleware; and interfaces to remote visualization environments. Work in the area of the sociology of knowledge sharing and management indicate that useful information is exchanged through social activities that involve physical collocation such as the coffee bar or water cooler. Digital equivalents of these are being used with some success.

The use of extranets and intranets has certainly been one of the main success stories in applied knowledge management. Corporate intranets allow best practice to be disseminated, and enable rapid dissemination of content. Extranets enable the coordination and collaboration of virtual teams of individuals.

A wide variety of components exist to support knowledge working on the knowledge grid. There is still much to do in order to exploit potential synergies between these technologies. Moreover, there is a great deal of research needed to further develop tools for each of the major phases of the knowledge life cycle. There are many challenges to be overcome if we are to support the problem solving activities required to enact a knowledge grid for e-Scientists (see section 5.5 for more details).

5.4 Knowledge Layer Aspects of the Scenario

Let us now consider our scenario in terms of the opportunities it offers for knowledge technology services (see table 5.1). We will describe the knowledge layer aspects in terms of the agent-based service oriented analysis developed in section 2.3. Important components of section 2.3 were the software proxies for human agents such as the *scientist agent* (SA) and a *technician agent* (TA). These software agents will interact with their human counterparts to elicit preferences, priorities and objectives. The software proxies will then realise these elicited items on the Grid. This calls for knowledge acquisition services. A range of methods could be used. Structured interview methods invoke templates of expected and anticipated information. Scaling and sorting methods enable humans to rank their preferences according to relevant attributes that can either be explicitly elicited or pre-enumerated. The laddering method enables users to construct or select from ontologies. Knowledge capture methods need not be explicit – a range of pattern detection and induction methods exist that can construct, for example, preferences from past usage.

One of the most pervasive knowledge services in our scenario is the partial or fully automated annotation of scientific data. Before it can be used as knowledge, we need to equip the data with meaning. Thus agents require capabilities that can take data streaming from instruments and annotate it with meaning and context. Example annotations include the experimental context of the data (where, when, what, why, which, how). Annotation may include links to other previously gathered information or its contribution and relevance to upcoming and planned work. Such knowledge services will certainly be one of the main functions required by our Analyser Agent and Analyser Database Agent (ADA). In the case of the High Resolution Analyser Agent (HRAA) we have the additional requirement to enrich a range of media types with annotations. In the original scenario this included video of the actual experimental runs.

These acquisition and annotation services along with many others will be underpinned by ontology services that maintain agreed vocabularies and conceptualizations of the scientific domain. These are the names and relations that hold between the objects and processes of interest to us. Ontology services will also manage the mapping between ontologies that will be required by agents with differing interests and perspectives.

Agent Requirements	Knowledge Technology Services
Scientist Agent (SA)	Knowledge Acquisition of Scientist Profile Ontology Service
Technician Agent (TA)	Knowledge Acquisition of Technician Profile Ontology Service Knowledge Based Scheduling Service to book analyser
Analyser Agent (AA)	Annotation and enrichment of instrument streams Ontology Service
Analyser Database Agent (ADA)	Annotation and enrichment of databases Ontology Service
High Resolution Analyser Agent (HRAA)	Annotation and enrichment of media Ontology Service Language Generation Services Internet Reasoning Services
Interest Notification Agent (INA)	Knowledge Publication Services Language Generation Services Knowledge Personalisation Services Ontology Service
Experimental Results Agent (ERA)	Language Generation Services Result Clustering and Taxonomy Formation Knowledge and Data Mining Service Ontology Service
Research Meeting Convener Agent (RMCA)	Constraint Based Scheduling Service Knowledge Personalisation Service Ontology Service
International Sample Database Agent (ISDA)	Result Clustering and Taxonomy Formation Knowledge and Data Mining Services Ontology Service
Paper Repository Agent (PRA)	Annotation and enrichment of papers Ontology Service Dynamic Link Service Discussion and Argumentation Service
Problem Solving Environment Agent (PSEA)	Knowledge Based Configuration of PSE Components Knowledge Based Parameter Setting and Input Selection Ontology Service

Table 5.1: Example knowledge technology services required by agents in the scenario

Personalisation services will also be invoked by a number of our agents in the scenario. These might interact with the annotation and ontology services already described so as to customize the generic annotations with personal markup – the fact that certain types of data are of special interest to a particular individual. Personal annotations might reflect genuine differences of terminology or perspective – particular signal types often have local vocabulary to describe them. Ensuring that certain types of content are noted as being of particular interest to particular individuals brings us on to services that notify and push content in the direction of interested parties. The Interest Notification Agent (INA) and the Research Meeting Convener Agent (RMCA) could both be involved in the publication of content either customized to individual or group interests. Portal technology can support the construction of dynamic content to assist the presentation of experimental results.

Agents such as the High Resolution Analyser (HRAA) and Experimental Results Analyser (ERA) have interests in classifying or grouping certain information and annotation types together. Examples might include all signals collected in a particular context, sets of signals collected and sampled across contexts. This in turn provides a basis for knowledge discovery and the mining of patterns in the content. Should such patterns arise these might be further classified against existing pattern types held in international databases – in our scenario this is managed in market places by agents such as the International Sample Database Agent (ISDA).

At this point agents are invoked whose job it is to locate other systems or agents that might have an interest in the results. Negotiating the conditions under which the results can be released, determining the quality of results, might all be undertaken by agents that are engaged to provide result brokering and result update services.

Raw results are unlikely to be especially interesting so that the generation of natural language summaries of results will be important for many of the agents in our scenario. Results that are published this way will also want to be linked and threaded to existing papers in the field and made available in ways that discussion groups can usefully comment on. Link services are one sort of knowledge technology that will be ubiquitous here – this is the dynamic linking of content in documents in such a way that multiple markups and hyperlink annotations can be simultaneously maintained. Issue tracking and design rationale methods allow multiple discussion threads to be constructed and followed through documents. In our scenario the Paper Respository Agent (PRA) will not only retrieve relevant papers but mark them up and thread them in ways that reflect the personal interests and conceptualizations (ontologies) of individuals or research groups.

The use of Problem Solving Environment Agents (PSEAs) in our simulation of experimentally derived results presents us with classic opportunities for knowledge intensive configuration and processing. Once again these results may be released to communities of varying size with their own interests and viewpoints.

Ultimately it will be up to application designers to determine if the knowledge services described in this scenario are invoked separately or else as part of the inherent competences of the agents described in section 2.3. Whatever the design decisions, it is clear that knowledge services will play a fundamental role in realizing the potential of the Semantic Grid for the e-Scientist.

5.5 Research Issues

The following is by no means an exhaustive list of the research issues that remain for exploiting knowledge services in the e-Science Grid. They are, however, likely to be the key ones. There are small-scale exemplars for most of these services. Consequently many of the issues relate to the problems of scale and distribution

1. Languages and infrastructures are needed to describe, advertise and locate knowledge level services. We need the means to invoke and communicate the results of such services. This is the sort of work that is currently underway in the Semantic Web effort of DAML-S and has been mentioned in section 4. However, it is far from clear how this work will interface with that of the agent based computing, Web Services and grid communities.
2. Methods are required to build large-scale ontologies and tools deployed to provide a range of ontology services.

3. Annotation services are required that will run over large corpora of local and distributed data. In some cases, for example, the annotation and cleaning of physics data, this process will be iterative and will need to be near real time as well as supporting fully automatic and mixed initiative modes. These annotation tools are required to work with mixed media.
4. Knowledge capture tools are needed that can be added as plugins to a wide variety of applications and which draw down on ontology services. This will include a clearer understanding of profiling individual and group e-Science perspectives and interests.
5. Dynamic linking, visualization, navigation and browsing of content from many perspectives over large content sets
6. Retrieval methods based on explicit annotations.
7. Construction of repositories of solution cases with sufficient annotation to promote reuse as opposed to discovering the solution again because the cost of finding the reusable solution is too high.
8. Deployment of routine natural language processing as Internet services. Capabilities urgently required include: tagging and markup of documents, discovering different linguistic forms of ontological elements, and providing language generation and summarization methods for routine scientific reporting
9. Deployment of Internet based reasoning services – whether as particular domain PSEs or more generic problem solvers such as scheduling and planning systems.
10. Provision of knowledge discovery services with standard input/output APIs to ontologically mapped data
11. Understanding how collaboration can be promoted using these knowledge services with technologies such as the Access Grid.
12. Understanding how to embed knowledge services in ubiquitous and pervasive devices

6. Research Agenda

The following general recommendations arise from our research and analysis of the state of the art and of the longer term vision for the Semantic Grid as outlined in this report. These recommendations also embody our sense of the most important issues that need to be addressed to effectively provide a computational infrastructure for e-Science in general and the provision of grid based computational support in particular. The recommendations are clustered into six themes and their ordering is not significant.

Technical and Conceptual Infrastructure

These recommendations relate both to the languages and architectures at a technical and a conceptual level.

1. Grid Toolkits - The technology and methodology does not exist to build the Semantic Grid today. However developments to this end are likely to be evolutionary and will need to include foundational elements found in widely used toolkits such as Globus. Within these toolkits there remain issues that still demand further research although it is important to be aware of what the US is doing in these areas and for the UK to play to its strengths. Issues that still need to be resolved are listed in section 3.6 and they include; naming, resource discovery, synchronisation, security, fault tolerance, dependability, integration of heterogeneous resources, scalability and performance.

2. Smart Laboratories - We believe that for e-Science to be successful and for the Grid to be effectively exploited much more attention needs to focused on how laboratories need to be instrumented and augmented. For example, infrastructure that allows a range of equipment to advertise its presence, be linked together, annotate and markup content it is receiving or producing. This should also extend to the use of portable devices and should include support for next generation Access Grids.

3. Service Oriented Architectures - Research the provision and implementation of e-Science and grid facilities in terms of service oriented architectures. Also research into service description languages as a way of describing and integrating the problem solving elements of an e-Science grid. Here we believe the emerging Web Services standards appear well suited to the e-Science infrastructure. Although these technologies have not yet fully emerged from the standards process, toolkits and test services exist and it is possible to build systems with these now.

4. Agent Based Approaches - Research the use of agent based architectures and interaction languages to enable e-Science marketplaces to be developed, enacted and maintained. We believe that such approaches provide a level of abstraction and define capabilities essential to realising the full potential of the Semantic Grid.

5. Network Philosophies – Research into the role of lightweight communication protocols for much of the threading of e-Science workflow. Investigate the likely evolution of various network and service distributions. For example, the extent to which within our computational networks there will be islands of high capacity grid clusters amounting to virtual private network grids. Investigate the merits of a range of fundamentally different configurations and architectures – for example, peer-to-peer, WebFlow and JINI. Research the relative merits of synchronous versus asynchronous approaches in our e-Science and grid contexts.

6. Trust and Provenance – Further research is needed to understand the processes, methods and techniques for establishing computational trust and determining the provenance and quality of content in e-Science and grid systems. This extends to the issue of digital rights management in making content available.

Content Infrastructure

These recommendations relate to the technologies and methods that are relevant to the way in which content is hosted and transacted on the Grid in e-Science contexts.

7. Metadata and Annotation – Whilst the basic metadata infrastructure already exists in the shape of RDF, metadata issues have not been fully addressed in current grid deployments. It is relatively straightforward to deploy some of the technology in this area, and this should be promoted. RDF, for example, is already encoding metadata and annotations as shared vocabularies or ontologies. However, there is still a need for extensive work in the area of tools and methods to support the design and deployment of e-Science ontologies. Annotation tools and methods need to be developed so that emerging metadata and ontologies can be applied to the large amount of content that will be present in the Grid and e-Science applications.

8. Knowledge Technologies – In addition to the requirement for the research in metadata and annotation above, there is a need for a range of other knowledge technologies to be developed and customised for use in e-Science contexts. These are described in detail in section 5.5 and include knowledge capture tools and methods, dynamic content linking, annotation based search, annotated reuse repositories, natural language processing methods (for content tagging, mark-up, generation and summarisation), data mining, machine learning and internet reasoning services. These technologies will need shared ontologies and service description languages if they are to be integrated into the e-Science workflow. These technologies will also need to be incorporated into the pervasive devices and smart laboratory contexts that will emerge in e-Science.

9. Integrated Media – Research into incorporating a wide range of media into the e-Science infrastructure. This will include video, audio, and a wide range of imaging methods. Research is also needed into the association of metadata and annotation with these various media forms.

10. Content Presentation – Research is required into methods and techniques that allow content to be visualised in ways consistent with the e-Science collaborative effort. This will also involve customising content in ways that reflect localised context and should allow for personalisation and adaptation.

Bootstrapping Activities

These recommendations relate to the processes and activities that are needed to get the UK's Grid and e-Science infrastructure more widely disseminated and exemplified.

11. Starter Kits - Currently the services provided by the grid infrastructure are somewhat rudimentary, their functionality is changing and the interfaces to these services are evolving. Grids will not be used to their full potential until developers have access to services with stable and standard interfaces, as well as richer functionality. Moreover the take up of the Grid will not happen until there are tools and utilities that facilitate the development of grid-based applications. Thus there is a clear need for more comprehensive starter kits to include many more tutorial examples of what can be achieved. To this end, we recommend continued research into development and deployment of portal technology and capabilities to provide access to grid resources in an intuitive and straightforward fashion

12. Exemplar and Reference Sites – Related to 11 above, there are a whole raft of problems and issues associated with the take up and use of grid concepts, infrastructure, and applications. Grids will not be used widely and successfully until there is a fully functional grid infrastructure (this includes middleware and tools to help developers of grid-based applications) and the grid infrastructure will not mature and stabilise until it has been fully tested by a whole range of varying kinds of grid-applications. However, the establishment, documentation and dissemination of exemplar sites and applications should be undertaken as a matter of urgency.

13. Use Cases - Grid software and experience to date primarily relate to contexts where there are relatively few nodes but these nodes have large internal complexity. We need to analyse current best practice and develop use cases to establish the strengths and weaknesses of these approaches. This should include opportunities that can be seen for additional services and gap analysis to determine what is obviously missing. A careful understanding of the medium scale heterogeneous IPG at NASA would be a useful initial example.

Human Resource Issues

These recommendations relate specifically to potential bottlenecks in the human resources needed to make a success of the UK e-Science and grid effort.

14. Community Building – There is a clear need for the UK e-Science and grid developers to establish strong links and watching briefs with the following technical communities – Semantic Web and Web Services. There is also a need to develop a balanced relationship between the application scientists and computer scientists. Here multi- and inter- disciplinarity are key requirements.

15. System Support - Discussions and efforts related to the Grid and e-Science are usually centred around computer and application scientists. Comparatively little attention is paid to system administrators, who need to install, set up, and manage these new wide-area environments. It is important that tools and utilities to ease the burden of the maintainers of grid-based infrastructure and applications are produced in parallel with other grid software and that this constituency is well represented in efforts to community build that arise out of 14.

16. Training – It is important that dissemination and training of the current and next generation of computer and application scientists, and system administrators is built into the evolving UK e-Science effort.

e-Science Intrinsics

These recommendations are concerned with obtaining a better understanding of the actual processes and practice of e-Science.

17. e-Science Workflow and Collaboration - Much more needs to be done to understand the workflow of current and future e-Science collaborations. Users should be able to form, maintain and disband communities of practice with restricted membership criteria and rules of operation. Currently most studies focus on the e-Science infrastructure *behind* the socket on the wall. However this infrastructure will not be used unless it fits in with the working environment of the e-Scientists. This process has not been studied explicitly and there is a pressing need to gather and understand these requirements. There is a need to collect real requirements from users, to collect use cases and to engage in some evaluative and comparative work. There is also a need to understand the process of collaboration in e-Science in order to fully and accurately define requirements for next generation Access Grids.

18. Pervasive e-Science - Currently most references and discussions about grids imply that their primary task is to enable global access to huge amounts of computational power. Generically, however, we believe grids should be thought of as the means of providing seamless and transparent access from and to a diverse set of networked resources. These resources can range from PDAs to supercomputers and from sensor’s and smart laboratories to satellite feeds.

Future Proposed Directions

These recommendations relate to strategic activities that need to be undertaken in order to maximise the leverage from the Semantic Grid endeavours.

19. Core Computer Science Research – If the full potential of the Semantic Grid is to be realised then a research program addressing the core computer science research issues identified in this report needs to be established. Without such research, the grid applications that are developed will be unable to support the e-Scientist to the degree that is necessary for e-Science technologies to be widely deployed and exploited.

20. e-Anything – Many of the issues, technologies and solutions developed in the context of e-Science can be exploited in other domains where groups of diverse stakeholders need to come together electronically and interact in flexible ways. Thus it is important that relationships are established and exploitation routes are explored with domains such as e-Business, e-Commerce, e-Education, and e-Entertainment.

7 References

[AccessGrid] Access Grid, http://www-fp.mcs.anl.gov/fl/accessgrid/

[Akarsu98] E. Akarsu, G.C. Fox, W. Furmanski, and T. Haupt, "WebFlow: High-Level Programming Environment and Visual Authoring Toolkit for High Performance Distributed Computing", SC98: High Performance Networking and Computing, Orlando, Florida, 1998.

[Barrett98] Barrett, R. and Maglio, P. P. "Intermediaries: New places for producing and manipulating web content", Proc. of the Seventh International World Wide Web Conference (WWW7), Brisbane, Australia. 1998.

[BernersLee01] Berners-Lee,T., Hendler,J. and Lassila ,O. "The Semantic Web", Scientific American, May 2001.

[BernersLee99] Berners-Lee, T. with Fischetti, M. "Weaving the Web:The Original Design and Ultimate Destiny of the World Wide Web by its Inventor", Harper, San Francisco, 1999.

[Bolosky00] W. J. Bolosky, J. R. Douceur, D. Ely, M. Theimer "Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs", Proc. international conference on measurement and modeling of computer systems (SIGMETRICS 2000), pp. 34-43. ACM Press, 2000.

[Buyya00a] Buyya R, Abramson D, and Giddy J, "Economy Driven Resource Management Architecture for Computational Power Grids", Proc. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), Las Vegas, USA, 2000.

[Buyya00b] Buyya R, Abramson D, and Giddy J "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid", Proc. 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia'2000), Beijing, China. IEEE Computer Society Press, USA, 2000.

[CCA] Common Component Architecture Forum (CCA Forum), http://www.acl.lanl.gov/cca-forum/

[Cerf93] V.G. Cerf et al., "National Collaboratories: Applying Information Technologies for Scientific Research", National Academy Press: Washington, D.C., 1993.

[Clark01a] D. Clark, "Face-to-Face with Peer-to-Peer Networking", Computer, Vol. 34, No. 1, January 2001, pp. 18-21

[Clarke01b], Clarke, I., Sandberg, O., Wiley, B. and Hong, T.W., "Freenet: A Distributed Anonymous Information Storage and Retrieval System", in ICSI Workshop on Design Issues in Anonymity and Unobservability, 2001.

[Coen98] M.A.Coen, "A Prototype Intelligent Environment," Cooperative Buildings – Integrating Information, Organisation and Architecture, N. Streitz, S.Konomi and H-J.Burkhardt (eds), LNCS, Springer-Verlag, 1998.

[Condor] Condor, http://www.cs.wisc.edu/condor/

[DataGrid] DataGrid project, http://www.eu-datagrid.org

[Davis99] Hugh C. Davis and David E. Millard and Sigi Reich and N. Bouvin and K. Grønbæk and P. J. Nürnberg and L. Sloth et al. “Interoperability between Hypermedia Systems: The Standardisation Work of the OHSWG”, Proc 10th ACM Conference on Hypertext and Hypermedia, Darmstadt, February 21-25, pp.201-202. 1999.

[Decker98] Decker, S., Brickley, D., Saarela, J. and Angele, J.A. “A query and inference service for RDF” Proceedings of QL'98: The Query Languages Workshop.

[Decker99] Decker, S., Erdmann, M., Fensel, D. and Studer, R. “Ontobroker: ontology-based access to distributed and semi-structured information” in R. Meersman (ed.) Semantic Issues in Multimedia Systems: Proceedings of DS-8, Kluwer Academic, Boston, 1999, 351-369.

[Druschel01] P. Druschel and A. Rowstron, “PAST: A large-scale, persistent peer-to-peer storage utility”, HotOS VIII, Schoss Elmau, Germany, May 2001.

[Englebart62] D. Englebart “Augmenting Human Intellect: A Conceptual Framework” AFOSR-3233, Oct. 1962. http://sloan.stanford.edu/mousesite/EngelbartPapers/ B5_F18_ConceptFrameworkInd.html

[Entropia] Entropia, http://entropia.com/

[EQUATOR] Equator IRC project http://www.equator.ac.uk/

[EuroGrid] EuroGrid, http://www.eurogrid.org/

[Faratin99] P. Faratin, C. Sierra, and N. R. Jennings, “Negotiation decision functions for autonomous agents” *Int. J. of Robotics and Autonomous Systems* **24** (3-4) 159-182. 1999.

[FIPA] The Foundation for Physical Agents http://www.fipa.org/

[Foster01] I. Foster, C. Kesselman, S. Tuecke “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, International Journal of Supercomputer Applications and High Performance Computing, 2001.

[Foster97a] I. Foster, J. Geisler, W. Nickless, W. Smith, S. Tuecke “Software Infrastructure for the I-WAY High Performance Distributed Computing Experiment” in Proc. 5th IEEE Symposium on High Performance Distributed Computing. pp. 562-571, 1997.

[Foster97b] I. Foster and C. Kesselman, “Globus: A Metacomputing Infrastructure Toolkit”, International Journal of Supercomputer Applications, 11(2): 115-128, 1997.

[Foster98] Ian Foster and Carl Kesselman (eds), “The Grid: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann, July 1998. ISBN 1-55860-475-8.

[Foster99] I. Foster, J. Insley, G. vonLaszewski, C. Kesselman, M. Thiebaux, “Distance Visualization: Data Exploration on the Grid”, IEEE Computer Magazine, 32 (12):36-43, 1999

[Fu00] Kevin Fu, M. Frans Kaashoek, and David Mazières “Fast and secure distributed read-only file system” in the Proceedings of the 4th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2000), San Diego, California, October 2000.

[Gannon99] D. Gannon and A. Grimshaw (1999) “Object-based approaches” in The Grid (eds I. Foster and C. Kesselman) Morgan Kaufmann.

[Geodise] http://www.geodise.org/

[GGF] Global Grid Forum – http://www.gridforum.org/

[Globus] Globus - http://www.globus.org/

[Gnutella] Gnutella, http://www.gnutella.co.uk/

[GridPort] SDSC GridPort Toolkit, http://gridport.npaci.edu/

[Grimshaw97] Grimshaw A., Wulf W. et al., “The Legion Vision of a Worldwide Virtual Computer”. Communications of the ACM, vol. 40(1), January 1997.

[GriPhyN] GriPhyN, Grid Physics Network, http://www.griphyn.org/

[Haber91] Haber, S and Stornetta, WS, “How to time-stamp a digital document”, Journal of Cryptography, Vol 3 No 2 pp 99-111, 1991.

[Haupt99] T. Haupt, E. Akarsu, G. Fox and W Furmanski, “Web Based Metacomputing”, Special Issue on Metacomputing, Future Generation Computer Systems, North Holland 1999.

[Hendler00] J. Hendler and D. McGuinness, “The DARPA Agent Markup Language,” IEEE Intelligent Systems, vol. 15, no. 6, Nov./Dec. 2000, pp. 72–73.

[Hoffman95] Hoffman, R., Shadbolt, N.R., Burton, A.M. and Klein,G. (1995) “Eliciting Knowledge from Experts: A Methodological Analysis” *Organizational Behavior and Decision Processes* vol.62(2) (1995) pp.129-158. Academic Press 0749-5978/95.

[Horrocks99] I. Horrocks and P.F. Patel-Schneider, “Optimizing Description Logic Subsumption,” *J. Logic and Computatio*n, vol. 9, no. 3, June 1999, pp. 267–293.

[HotPage] HotPage, https://hotpage.npaci.edu/

[HTC] HTC, http://www.cs.wisc.edu/condor/htc.html

[IPG] NASA Information Power Grid, http://www.ipg.nasa.gov/

[IPv6] IPv6 Forum, http://www.ipv6forum.com/

[JavaGrande] Java Grande, http://www.javagrande.org/

[Jennings00] N. R. Jennings, “On agent-based software engineering”, Artificial Intelligence 117 277-296. 2000.

[Jennings01] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra and M. Wooldridge, “Automated Negotiation: Prospects, Methods and Challenges” Int Journal of Group Decision and Negotiation 10(2) 199-215. 2001.

[Jini] JINI, http://www.jini.org

[JXTA] JXTA, http://www.jxta.org/

[Kraus01] S. Kraus “Strategic Negotiation in Multi-agent Environments” MIT Press. 2001.

[Kubiatowicz00] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. “OceanStore: An Architecture for Global-Scale Persistent Storage”, in Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), November 2000.

[Legion] Legion, http://legion.virginia.edu/

[Leigh99] J. Leigh et al. “A Review of Tele-Immersive Applications in the CAVE Research Network”, Proceedings of the IEEE Virtual Reality 2000 International Conference (VR 2000), 1999.

[Luke00] S. Luke and J. Heflin, “SHOE 1.01. Proposed Specification”, www.cs.umd.edu/projects/plus/SHOE/spec1.01.html, 2000 (current 20 Mar. 2001).

[McIlraith01] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng “Semantic Web Services” IEEE Intelligent Systems, March/April 2001, Vol 16, No 2, pp. 46-53.

[Milton99] Milton, N., Shadbolt, N., Cottam, H. & Hammersley, M. (1999). “Towards a Knowledge Technology for Knowledge Management” *International Journal of Human-Computer Studies*, 51(3), 615-64

[MPEG21] ISO/IEC JTC1/SC29/WG11 Coding of Moving Picture and Audio, MPEG-21 Overview, document N4041.

[Napster] Napster, http://www.napster.com/

[Newell82] A. Newell, “The Knowledge Level” Artificial Intelligence 18 87-127.

[Nimrod] Nimrod/G, http://www.dgs.monash.edu.au/~davida/nimrod.html

[NLANR] NLANR Grid Portal Development Kit, http://dast.nlanr.net/Features/GridPortal/

[OMG] OMG, http://www.omg.org

[Page01] Kevin R. Page, Don Cruickshank and David De Roure. It’s About Time: Link Streams as Continuous Metadata. Proc The Twelfth ACM Conference on Hypertext and Hypermedia (Hypertext '01) p.93-102. 2001.

[Parabon], Parabon, http://www.parabon.com/

[PBS] Portable Batch System, http://www.openpbs.org/

[Pentland96] Pentland “Smart Rooms” Scientific American 274, No. 4, pp 68-76, April 1996.

[PPG] Particle Physics Data Grid - http://www.cacr.caltech.edu/ppdg/

[Probity] “Public Registration On the Web of Intellectual property”, http://www.probity.org/

[Sagonas94] K. Sagonas, T. Swift, and D. Warren, “XSB as an Efficient Deductive Database Engine,” Proc. 1994 ACM SIGMOD Int’l Conf. Management of Data (SIGMOD’94), ACM Press, New York, 1994, pp. 442–453.

[Sandholm00] T. Sandholm “Agents in Electronic Commerce: Component Technologies for Automated Negotiation and Coalition Formation” Autonomous Agents and Multi-Agent Systems, 3(1), 73-96. 2000.

[Schrieber00] Schreiber G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N.R, Van de Velde, W. and Wielinga, B. (2000) *Knowledge Engineering and Management.* MIT Press.

[Semweb] W3C Semantic Web Activity Statement - http://www.w3.org/2001/sw/Activity

[SETI] SETI@Home, http://setiathome.ssl.berkeley.edu/

[SGE] Sun Grid Engine, http://www.sun.com/software/gridware/

[Shehory98] O. Shehory and S. Kraus "Methods for task allocation via agent coalition formation" Artificial Intelligence, 101 (1-2) 165-200. 1998.

[SRB] Storage Resource Broker, http://www.sdsc.edu/DICE/SRB/

[Staab00] Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H.-P., Studer, R. and Sure, Y. "Semantic community Web portals", in Proceedings of WWW-9, Amsterdam, 2000, http://www9.org/w9cdrom/134/134.html.

[Stoffel97] K. Stoffel, M. Taylor, and J. Hendler, "Efficient Management of Very Large Ontologies," Proc. 14th Nat'l Conf. AI, MIT–AAAI Press, Menlo Park, Calif., 1997.

[UDDI] http://www.uddi.org/

[UNICORE] http://www.unicore.de/

[vanHarmelen00] F. van Harmelen and I. Horrocks, "FAQs on OIL: The Ontology Inference Layer," IEEE Intelligent Systems, vol. 15, no. 6, Nov./Dec. 2000, pp. 69–72.

[WebDAV] Web-based Distributed Authoring and Versioning, http://www.Webdav.org/

[WebServices01] Proceedings of W3C Web Services Workshop, April 11-12, 2001. http://www.w3.org/2001/03/wsws-program

[Wooldridge97] M. Wooldridge, "Agent-based software engineering". IEE Proc on Software Engineering 144 (1) 26-37. 1997.

[WSDL] Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001 http://www.w3.org/TR/wsdl

[WSFL] Web Services Flow Language (WSFL) Version 1.0, http://www-4.ibm.com/software/solutions/Webservices/pdf/WSFL.pdf

[XLANG] Web Services for Business Process Design, http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

[XMLP] XML Protocol Activity, http://www.w3.org/2000/xp/

[Zhuang01] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz and John Kubiatowicz. "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination", Proceedings of the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001), June 2001.