On the Use of Agents in a BioInformatics Grid

Luc Moreau[1], Simon Miles[1], Carole Goble[2], Mark Greenwood[2], Vijay Dialani[1], Matthew Addis[1], Nedim Alpdemir[2], Rich Cawley[2], David De Roure[1], Justin Ferris[1], Rob Gaizauskas[6], Kevin Glover[3], Chris Greenhalgh[3], Peter Li[5], Xiaojian Liu[1], Phillip Lord[2], Michael Luck[1], Darren Marvin[1], Tom Oinn[4], Norman Paton[2], Stephen Pettifer[2], Milena V Radenkovic[3], Angus Roberts[2], Alan Robinson[4], Tom Rodden[3], Martin Senger[4], Nick Sharman[2], Robert Stevens[2], Brian Warboys[2], Anil Wipat[5], Chris Wroe[2]

(1) University of Southampton, (2) University of Manchester, (3) University of Nottingham, (4) EMBL Outstation - European Bioinformatics Institute, (5) University of Newcastle, (6) University of Sheffield

Project site: `www.mygrid.org.uk`
Contact Author: Luc Moreau (`L.Moreau@ecs.soton.ac.uk`)

Abstract

MyGrid is an e-Science Grid project that aims to help biologists and bioinformaticians to perform workflow-based in silico *experiments, and help them to automate the management of such workflows through personalisation, notification of change and publication of experiments. In this paper, we describe the architecture of myGrid and how it will be used by the scientist. We then show how myGrid can benefit from agents technologies. We have identified three key uses of agent technologies in myGrid:* user agents, *able to customize and personalise data,* agent communication languages *offering a generic and portable communication medium, and* negotiation *allowing multiple distributed entities to reach service level agreements.*

1 Introduction

MyGrid is a Grid middleware project in a bioinformatics setting. In biological sciences, it is not principally the size of the data that matters but the complexity involved in using it: the complexity of the data itself, the number of repositories and tools that need to be involved in the computations required to answer the kind of questions posed by the scientist, and the heterogeneity of the data and operation of tools. Rather than a few international facilities (e.g. CERN and Fermi Lab) producing vast amounts of data that need to be accessible, the pressing issue with bioinformatics is coping with a very large number of sites (potentially thousands of individual laboratories) around the world, each using cheap, commodity technology to continuously generate substantial quantities of different kinds of data, and design new tools to process it.

In many resources, each record is analogous to an individual publication with not only raw data, but also additional annotations supplied by a small number of human experts (curators) or automated systems. Annotations are typically semi-structured text that make some use of keywords and controlled vocabularies, and have to be parsed computationally or read by people. Therefore, as well as a large number of data types, much of the valuable knowledge is locked into semi-structured text, under the premise that the scientist will read and interpret it.

In the past, this complexity has been dealt with largely by the intelligence of the practising biologist. This has been possible because biologists working on a specific organism, or a specific aspect of it, have needed access to only a small number of these resources.

Interestingly, much of the growth of molecular biology has been contemporaneous with the development of the Web, which probably explains why many resources have been designed with the intention that a scientist will interact with a Web page, dealing with a single query at a time, and read the results displayed as reports in a browser, navigating between links in different databases by mouse-clicking. We call this approach "query by navigation". Where databases are published, they are usually released as flat files, even in those cases, such as SWISS-PROT and EMBL [15], where the production systems are relational databases.

Although the volume of data is not yet a computational problem, the advent of high throughput experiment techniques means that human analysis is now reaching its limits. With sequence databases reaching hundreds of MBytes and microarray expression data producing tens of GBytes, the limits of the non-scalable query by navigation are rapidly being reached, if not already passed.

The particular focus of myGrid, therefore, is on increasingly data-intensive bioinformatics and the provision of a distributed environment that supports the *in silico* experimental process. The vision is of a "lab book" environment where the e-Scientist can construct *in silico* experiments, and find and adapt others, store partial results in local data repositories and have their own view on public repositories, and be better informed as to the provenance and the currency of the tools and data directly relevant to their experimental space. For a less skilled user, myGrid should help in finding appropriate resources, offering alternatives to busy resources and guiding the user through the composition of resources into complex workflows. In order to provide such an environment, myGrid unequivocally needs to address the "Grid problem", i.e. the flexible, secure, coordinated resource sharing, among dynamic collection of individuals and institutions — *Virtual Organisations* [9]. In this context, the Grid becomes egocentrically based around the Scientist: *myGrid*.

The contributions of this paper are threefold. First, we present a service-based architecture to support the vision of the "lab book" environment. Second, we illustrate how this architecture can be used during the enactment of workflows. Third, we review how a bioinformatics grid can benefit from agent technologies.

2 The MyGrid Service-Oriented Architecture

In this section, we describe the different services that are provided by myGrid, and sketch their interactions. (They are displayed in Figure 1.) The experimental *in silico* process is expressed as a workflow script by the scientist. Services can be viewed as being provided by agents and workflow can be seen as an agent interaction script. Some initial work in this vein has already been done [2, 3].

2.1 Workflow Enactment

At the heart of the myGrid runtime system, we find the workflow enactment engine which, given a workflow script, is able to execute (or enact) the script. Scientists and their institutions may have preferences that must be taken into account when enacting a workflow script: e.g., some databases are preferred over others, or specific tools and parameters are routinely chosen. It is the role of the workflow resolution service to customise a script's "free variables", possibly making use of a workflow personalisation service able to obtain preferences from a user (or a user agent acting on their behalf). There exist several strategies to resolve a workflow: eagerly before enactment, or lazily if and when required by the enactment engine. (Both can be expressed at the level of scripts through the use of an appropriate program transformation.)

The workflow enactment engine can send requests to existing running services or can activate tools and interact with them: services need to be discovered and processes need to be created. For the former, a service directory is used as a repository of service instances that are currently active, whereas the latter makes use of a job activation and scheduling system. Generally, scripts may require space to store temporary results, or may try to ensure that computational resources are reserved at the same time as storage space to ensure the prompt execution of the workflow: allocation and reservation are handled by the resource management service.

2.2 User Interaction

The user, through an interface, may interact with the workflow enactment engine, suspending and resuming workflows, observing their progress, analysing their logs. Suspended workflows are serialised and stored in a repository, potentially shared with other users.

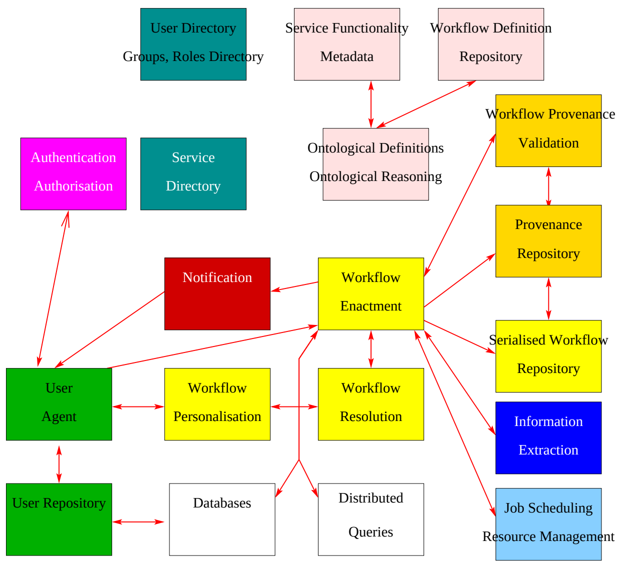


Figure 1. myGrid Services

Some workflows may take days, if not weeks, to complete their execution. Users therefore need to be notified when workflow execution terminates. We prefer not to assume the existence of user agents able to handle incoming notifications. Indeed, users are not logged on permanently, and we feel that always running user agents would overload the system unnecessarily. Instead, we make use of a notification service able to forward messages to user agents, when present, or to store messages in their absence. The use of the notification service is of course not restricted to the user agent, but may be used by any services in myGrid. In particular, we use the notification service to give notifications to users about changes in data, workflows, services.

Sharing information between users, discovering information, finding out users or institutions that are investigating given topics are all key functionalities of myGrid. Several directories are used for that purpose: the user directory holds information about users, groups, roles and institutions; the workflow repository contains information about scripts and their functionality.

2.3 Ontologies

All information about workflows and users is what we call *metadata* and is structured according to a set of ontologies — an *ontology* is generally defined as a shared understanding of a specific domain [10]. Information about services are also expressed using such ontologies, and are stored in the *service functionality metadata service*; the latter service contains metadata about classes of services, and must be distinguished from the *service directory* which lists active service instances.

Not only are ontologies a shared understanding of some domains, but their logical foundations also allow users to perform reasoning over such domains. Examples of reasoning include classification (i.e., the computation of a concept hierarchy based on the specialisation relation), or consistency checking (i.e., checking that a statement is not inconsistent in a logic). An ontology-based reasoning facility is provided by myGrid to help users compose new workflows. Additionally, the ontology service allows users to reason about concepts of the application domain in order to understand their inter-relationships.

2.4 Data and Metadata

Most myGrid repositories are be implemented as databases. Additionally, biological information is stored in multiple and heterogeneous databases. Distributed query systems over such databases are an essential component to facilitate information integration. In myGrid, databases is accessed though a service interface [16], whereby structured data stores support consistent interfaces for database access, manipulation and metadata description. As a component within the personalisation framework of myGrid, database services are used to provide individual users with access to *(i)* locally produced data sets; *(ii)* the results of analyses run by the user over local or remote data; and *(iii)* distributed querying over local and remote data resources. The distributed query processor benefits from the consistent service interfaces and metadata descriptions provided by local and remote databases.

Above, we have discussed the existence of metadata that is structured according to ontologies. In biological sciences, it is also customary to create annotations in free text form. Such metadata contains invaluable information assembled by database curators. MyGrid also provides support for correlating such an information with medical literature through an information extraction service.

MyGrid provides support for provenance in two different ways. First, provenance information, in particular related to workflow enactement, can be logged in the provenance annotation service; such a service is also used to store provenance information for services having no built-in support for provenance. Additionally, the workflow provenance validation service is able to re-enact workflows to establish change over time.

2.5 Security and Fault Tolerance

The myGrid authentication service extends the PKI infrastructure to provide X.509 certificates for users and objects (called *identities* henceforth) needing verification. It supports a notion of *logical domain* which is defined by the set of identities it manages. The confederation of several logical domains forms an *enterprise* infrastructure. Each logical domain has associated domain administrators who are authorised to create and revoke identities within their logical domains.

In myGrid, a sub-component of the user agent acts as a credentials repository, permitting simultaneous access to multiple logical domains. This facility allows a user to have simultaneous access to multiple virtual organisations [9] and obtain the access rights to multiple resources across sites.

MyGrid supports role-based access control [17] and dynamic mapping between users and roles. Within each logical domain, there exists a hierarchy of user roles and access rights; roles are statically associated with access rights. The model is extensible by allowing the definition of new roles and access rights. In an enterprise security infrastructure, one needs to support identities from different logical domains, which may have different access models: this requires the definition of a mapping of roles and access rights of a domain onto roles and access rights of another domain.

MyGrid computations may be long-lived and involve a very large number of computing resources. Hence, they need to be designed with fault tolerance in order to be robust. To this end, myGrid provides a set of interfaces, which services are required to implement, and which provide robustness to applications involving the use of multiple services. The complete description is beyond the scope of this paper, and we refer the reader to a companion paper [4]. The approach may be summarised as follows: implementors of a service have to implement an interface (for checkpoint and rollback); the architecture dynamically extends the service interface by methods for fault tolerance; applications making use of different services have to declare their inter-dependencies, which are used by a fault-manager to control checkpoints and rollbacks; an extension of the communication layer is able to log and replay messages.

3 MyGrid Workflow Enactment in Practice

We have implemented a prototype of this architecture, based on a subset of the services described in Figure 1 and exclusively relying on Web Services technology. In this section, we show how the scientist is able to enact workflows in myGrid.

An *in silico* experiment typically involves using several bioinformatics databases and algorithms available on the World Wide Web. Currently, these resources are integrated by a "query by navigation" process, i.e. by cutting and pasting across browser windows. Alternatively, a script (such as perl script or bat file) may be written to facilitate the frequent repetition of *in silico* experiments. There are a number of limitations of this current state of practice that workflows in the myGrid environment address.

First, there is the problem of knowing what *in silico* experiment to perform. A user typically has an understanding of what they are trying to achieve in bioinformatics terms and might know some specific Web resources or script, based on past experience. How they acquired this experience, how they keep their knowledge up-to-date, and how they adapt previous experiences to new tasks are essential elements of the experimental process, which we intend to make explicit.

Second, there is the problem of incorporating new resources. In most situations the user is interested in a specific type of resource, a SWISS-PROT database, rather than a specific resource instance such as the SWISS-PROT database hosted at a specific institution. If their first (default) choice is unavailable, then the user would like to use an alternative of the same type. In the current state of practice, scripts tend to include hard-coded references to specific resources.

Third, there is the limited recording of how *in silico* experiments have been performed. Without knowing what resources have been used in the derivation of a result, there is no way of knowing if it might be worthwhile re-running the *in silico* experiment in the light of more recent knowledge (or if the result should be disregarded, as more recent knowledge has rendered some of the experimental assumptions invalid.)

Fourth, there is difficulty in propagating good *in silico* experimental practice. This essentially incorporates the previous three issues and extends them beyond the individual scientist to the sharing of resources between research communities. Within an e-Science community, it is not just the available data that is valuable, but also knowing the acceptable/proven ways of combining that data to generate new insights.

3.1 Prototype Experiment

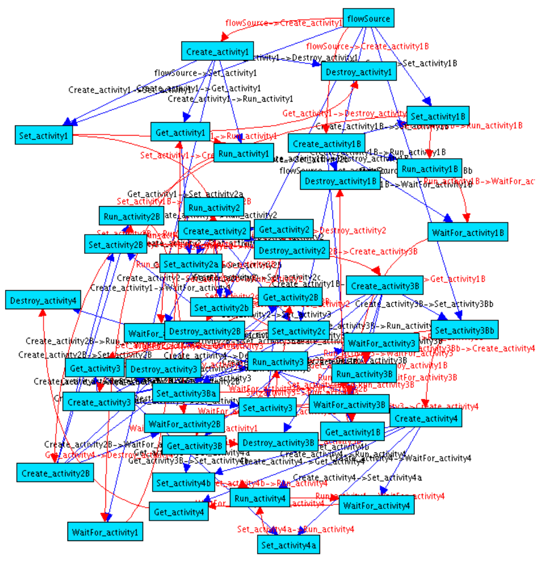
In our prototype, a myGrid user has access to a personal repository containing their domain data (and results), a workflow repository containing the available workflow scripts and a service directory of the available service instances. Each data item in the personal repository has an associated concept type (a term in the ontology); such concept types are used to initiate the enactment of *in silico* experiments, as we now explain.

Potential workflows are identified through a conversation with the ontology service. A specific user interface is used to incrementally build up an abstract description of a workflow, starting with the selected concept type. Once the abstract workflow description is complete, it can be classified to give a workflow service type identifier (also a term in the ontology). This is used to retrieve the identifiers of workflow scripts that match this required type, and from the identifier, the workflow script itself. In this way, the user interacts with the ontology service to determine the concepts that match their task; then, they get a list of all the workflow scripts of this type and choose the one to run (perhaps using some metadata to help in the selection).

3.2 Workflow Details

Inspired by WSFL [11], the workflow definition consists of a set of service providers, activities, data links and control links between activities (cf. Figure 2). For many myGrid workflows, each activity has its own service provider, which includes a locator element to identify the Web Service, to be used by the workflow enactment engine. It is possible for the locator to be static and directly reference the WSDL definition of the service, but it is more usual for the locator to be dynamic. In this case, it gives the service type identifier that is used to lookup possible services (using UDDI) from the service directory. Each activity is described in terms of its service provider and an operation, thus expressing the specific provided operation that matches the abstract activity in the workflow. The data links describe how the outputs of an activity are mapped to the inputs of other activities, while the control links are used to decide when the activities should be fired.

The enactment of a workflow script starts by sending the script and input data to the workflow enactment service. This responds by returning a workflow instance identifier that the user interface portal can use to query the workflow



```
<serviceProvider name="reformatting_seqret"
                 type="ebins:seqret_derived">
    <locator type="static"
             service="url of file.wsdl"/>
</serviceProvider>

<activity name="Run_activity1">
 <performedBy
     serviceProvider="reformatting_seqret"/>
 <implement>
  <export>
   <target portType="ebins:reformatting__seqret"
         operation="run"/>
  </export>
 </implement>
</activity>

<dataLink source="Create_activity1"
        target="Run_activity1">
   <map sourceMessage="createEmptyJobResponse"
        targetMessage="runRequest">
    <partMap source="return" target="in0"/>
   </map>
</dataLink>
```

Figure 2. Example of Workflow

status and identify the workflow result in the personal repository.

The use of a dynamic locator to identify a service provider in the workflow script is the main mechanism for abstracting a workflow over specific service instances. The dynamic locator gives the service type identifier; any service instance that has registered under this identifier in the service directory is a potential match. The dynamic locator also gives the policy to be used for selecting between the potential services. In the prototype, only two policies are available. In the simplest policy, the enactment engine chooses the first element in the list returned from the service directory. The other is user-choice, where the list of services is sent to the user agent who makes choice on behalf of the user, possibly interacting with the user through the portal, if configured to do so.

The workflow enactment service also creates a provenance log within the personal repository for each workflow instance. This trace includes: the initial data, the workflow script, the intermediate results, the actual service instances selected and the time taken for the service operations. These logs could be viewed through the portal to understand the detailed derivation of a particular result.

The definition of an *in silico* experiment as a workflow means that it exists as an explicit piece of data that can be shared, copied and altered by a community of scientists. Even within the context of the simple examples in the prototype, it was clear that what a user might consider a single *in silico* experiment might be supported by many workflows. There are variants of workflows that have the same type and the choice between them is often the personal choice of the user. Some users will always want to be involved in the dynamic selection between alternative services, while others will be content to leave that to the enactment engine, or an agent acting on their behalf. Another way that workflows of the same type might vary is in the filtering of sets of intermediate results. (In the current state of practice, this corresponds to users who do not mechanically cut and paste between resources, but apply their knowledge to select, cut and paste in their *in silico* experiments.)

While our project is still at an early stage, we have been able to enact workflows that expressed rather complex queries in bioinformatics, such as *(i)* Has anyone else studied the effect of neurotransmitters on the circadian rhythms of Drosophila? *(ii)* How do the functions of the clusters of proteins from my experiment interrelate? *(iii)* What are the proteins with a particular function? *(iv)* What is known about a given protein?

The enactment of workflows has shown that there is a need for user preferences to guide the selection of services to invoke. There is scope for user agents to (semi-)automate the customisation of service selection, and also for negotiation when multiple service with complementary characteristics are available to the user. This is precisely the role of *software agents*, which we discuss in the following section.

4 Agents in Bioinformatics Grids

The bioinformatics domain is characterised by rapid and substantial change over time. The volume of data poses problems, but the change in the resources available to the bioscientist is a distinct problem; new resources can appear, old ones can disappear, and some can simply change. Although there are several well-known and highly regarded databases, limiting a system to only these could impose undesirable constraints. Thus, any system intended for application to the bioinformatics domain should be able to cope with this dynamism and openness, and nothing addresses these concerns as comprehensively as the agent approach. Agents are flexible, autonomous components designed to undertake overarching strategic goals, while at the same time being able to respond to the uncertainty inherent in the environment. On the one hand, agents provide an appropriate paradigm or abstraction for the design of scalable systems aimed at this kind of problem; on the other, the field of agent-based computing offers a set of technologies that may be used for particular purposes in certain aspects of the system, including personalisation, communication, negotiation, which we discuss below.

4.1 User Agent

The user agent of Figure 1 is an agent in the sense that it *represents* a user within the myGrid system (so could also be described as a *personal agent* [12]). It can autonomously provide the personal preferences and conditions of a user to other parts of the system. This is useful, in particular, when a workflow is being enacted and a choice of services becomes available. The choice should not be made arbitrarily, but on the priorities and circumstances of the particular user. For example, a user may have greater trust in the ability of one service to produce accurate results than another, or the user's operating system may only support some forms of interaction between services and the user. The user should not have to be queried each time a service must be chosen, as these preferences and previous choices can be recorded and acted upon by the user agent to select from each set of options presented to it. We call this function *personalisation*.

Another application of the user agent is as a contact point between services within myGrid and the user. By having an intermediary able to receive, for example, requests from services for the user to enter data or notifications about changes to remote databases, these messages can be provided to the user only when the user is able and willing to receive them. Conversely, the user can delegate the details of a procedure to the user agent, such as authenticating itself with a service before use, or for personalisation of workflows as described above.

4.2 Agent Communication Language

A key requirement of myGrid is the design of a *future proof* environment in which collaborative distributed bioinformatics applications may be developed. Bioinformatics is not a green field, and multiple protocols and standards are already supported by the community. Our methodology is to design a *generic architecture* able to support multiple existing protocols, languages and standards, and which hopefully will be able to accommodate future developments. In particular, we want to design an *abstract communication architecture* that we can map onto concrete communication technologies.

At the same time, in the eBusiness community, Web Services have emerged as a set of open standards, defined by the World Wide Web consortium, and ubiquitously supported by IT suppliers and users. They rely on the syntactic framework XML, the transport layer SOAP [21], the XML-based language WSDL [20] to describe services, and the service directory UDDI [19]. Web Services therefore look like a strong contender for Grid Computing, as illustrated by the recent Open Grid Service Architecture (OGSA) [7] which extends Web Services with support for the dynamic lifecycle management of Grid Services.

The idea of an "agent communication language" dates back from the DARPA Knowledge Sharing Effort, which led to the design of KQML (Knowledge Query and Manipulation Language) [5], and was followed later by FIPA (Foundation for Intelligent Physical Agents) Agent Communication Language [6].

In agent systems, it is common practice to separate intention from content in communicative acts, abstracting and classifying the former according to Searle's speech act theory [18]. An agent's communications are thereby structured and classified according to a predefined set of "message categorisations", usually referred to as *performatives*.

In previous work, we have successfully adapted a key concept of the Nexus communication layer [8] to the world of agents, which resulted in SoFAR, the Southampton Framework for Agent Research [14]. Communications between agents take place over a *virtual communication link*, identified by a *startpoint* and an *endpoint*. An *endpoint* identifies an agent's ability to receive messages using a specific communication protocol. An endpoint extracts messages from the communication link and passes them onto the agent. A *startpoint* is the other end of the communication link, from which messages get sent to an endpoint. Given a startpoint, one can communicate with a remote agent, by activating a performative on the startpoint, passing the message content.

In [13, 1], we have described how the idea of agent communication languages, and the startpoint/endpoint communication model could be mapped onto the communication stack of Web Services. In [13], we only focused on the communication layer by encoding performatives and message contents in SOAP. In [1], we made use of the WSDL language to describe agents and the performatives they support, so that such definitions could be published in the UDDI registry, discovered and re-used like any other Web Service.

This approach turns out to be promising, as it offers a declarative communication semantics, which promotes inter-operability, openness, and dynamic discovery and reuse of agents. It also opens the agent world to the Web Services community, helping in the design of more complex interactions, as discussed in the following section.

4.3 Negotiation Broker

Another application of research from the agent field is in the area of negotiation. Service users and service providers typically have differing criteria over the preferable quality and content of the service they receive. An area in which negotiation can be seen as particularly useful in myGrid is *notification support*. The providers of various services may want to send out into the wider system notifications concerning improvements to tools, changes to databases or updates concerning the state of enacted workflows, etc. Other services or agents want to register to receive some subset of these notifications. For stability, we support asynchronous messages, and manage their distribution using a *notification service*.

4.3.1 Quality of Service

The subjects (quantitative and qualitative) over which negotiation takes place could include the following forms of *quality of service*:

- the cost of receiving the notification,
- the topic (event category) of the notifications,
- the frequency with which notifications are received (e.g. every time a change occurs, daily, hourly),
- the generality of the change described by the notifications,
- the form in which the information in the notification message is supplied, and
- the accuracy of information contained within a notification.

Quality of service refers to these distinctions in both what a publisher produces and how it produces it.

A publisher of notifications will be able to produce notifications matching (or exceeding, where appropriate) one or more measures of quality of service. For example, a publisher may be able to publish notifications on a particular topic every minute or every hour. A consumer of notifications may prefer, or demand, one measure of quality of service over another. Whether, or how well, their demands can be met by a publisher depends on the quality of service that the publisher can provide.

If demands cannot be met exactly, the consumer may choose to negotiate with the publisher to find the next best quality of service that the publisher can provide. For example, if the consumer desires notifications weekly and the publisher can provide daily or fortnightly notifications, the subscriber must find this out from the publisher and then decide between them, or decide not to subscribe at all, based on the subscribers particular priorities. Alternatively, the publisher may be able to exceed the quality of service in several ways which the subscriber may be unaware of, which could also lead to negotiation.

4.3.2 Model

As the notification service must provide notification support for a potentially large and varying number of consumers, it should not change its contract based solely on the results of negotiation between a single consumer and a publisher. Therefore, the notification service should have some control over the quality of service agreed upon. There are other reasons that the notification service may usefully limit the interaction between the publisher and consumer, such as limiting the knowledge of one by the other for reasons of privacy.

We propose using a *quality of service broker* that is an agent conceptually contained within the notification service (available through the same communication channels). The quality of service broker negotiates on behalf of each consumer wishing to receive notifications of a specified quality, then provide a final proposal to the consumer. It can negotiate with any of the publishers known to the notification service, and also limit the agreed quality of service to that acceptable to the notification service. We wish to make the quality of service broker able to negotiate with publishers produced by various providers, so we use the concept of *pluggable negotiation algorithms*, allowing the quality of service broker to select the appropriate protocol for negotiating with a publisher.

5 Conclusion

In this paper, we have presented the myGrid architecture and overviewed possible use of agents. MyGrid aims to provide a personalised environment for the bioscientists, which helps them to automate, repeat and therefore better achieve their experiments. Agents are particularly useful in tailoring the myGrid system to the priorities of individual scientists, personalising each step of a workflow and negotiating on their behalf. It can be seen from our discussion that, along with dynamic workflow enactment, standardisation of data semantics via ontologies and the many other facilities of myGrid, agents can make conducting in-silico experiments flexible and more easily controlled by the individual or collablorating scientists.

The examples of use of agents we have presented, while already offering a capability non existing in current bioinformatics environment, still remain rather localised to some specific services (user agent or negotiation over quality of service of notification service), or components such as a communication layer.

For the long term, agent-based computing also counts in its armoury a range of techniques for enabling individual components to collaborate with others, as well as for competing with others in the provision of services as may be found in bioinformatics. For example, the former aspects include issues in the construction of the virtual organisation mentioned earlier, whereby different services come together in some coherent whole subsystem for a particular purpose; and issues in the regulation of open societies of services through the use of norms and electronic institutions. The latter aspects, for example, include the possible use of sophisticated auction mechanisms, or electronic marketplaces, for obtaining the best services or resources at the least cost to the user. Additionally, whenever interactions take place between different agents, the issues of trust and reputation become important. Though some work has been done in this area, the focus on both agent-based computing and Grid computing has been limited, with the majority adopting the stance of assuming complete trust, and avoiding the issue; questions of deception and fraud in communication and interaction, of assurance and reputation, and of risk and confidence, are particularly significant, especially where interactions take place with new partners.

6 Acknowledgements

This research is funded by EPSRC myGrid project (reference GR/R67743/01).

References

[1] A. Avila-Rosas, L. Moreau, V. Dialani, S. Miles, and X. Liu. Agents for the Grid: A Comparison with Web Services (part II: Service Discovery). In *Workshop on Challenges in Open Agent Systems*, Bologna, Italy, July 2002.

[2] K. Bryson, M. Luck, M. Joy, and D. Jones. Agent interaction for bioinformatics data management. *Applied Artificial Intelligence*, 15(10):917–947, 2001.

[3] K. Decker, X. Zheng, and C. Schmidt. A Multi-Agent System for Automated Genetic Annotation. In *The fifth ACM International Conference on Autonomous Agents*, Montreal, Canada, May 2001.

[4] V. Dialani, S. Miles, L. Moreau, D. D. Roure, and M. Luck. Transparent fault tolerance for web services based architectures. In *Eighth International Europar Conference (EUROPAR'02)*, Lecture Notes in Computer Science, Padeborn, Germany, Aug. 2002. Springer-Verlag.

[5] T. Finin, Y. Labrou, and J. Mayfield. *Software Agents, J. Bradshaw, Ed.*, chapter KQML as an Agent Communication Language. MIT Press, 1997.

[6] FIPA: Foundation for Intelligent Physical Agents. `http://drogo.cselt.stet.it/fipa/`.

[7] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The Physiology of the Grid — An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Argonne National Laboratory, 2002.

[8] I. Foster, C. Kesselman, and S. Tuecke. The Nexus Approach to Integrating Multithreading and Communication. *Journal of Parallel and Distributed Computing*, 37:70–82, 1996.

[9] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid. Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 2001.

[10] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. Technical Report KSL-93-04, Knowledge Systems Laboratory, Stanford University, Aug. 1993.

[11] F. Leyman. Web Services Flow Language (WSFL). Technical report, IBM, May 2001.

[12] P. Maes. Agents that Reduce Work and Information Overload. *Commun. ACM*, 37(7):31–40, July 1994.

[13] L. Moreau. Agents for the Grid: A Comparison for Web Services (Part 1: the transport layer). In H. E. Bal, K.-P. Lohr, and A. Reinefeld, editors, *Second IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2002)*, pages 220–228, Berlin, Germany, May 2002. IEEE Computer Society.

[14] L. Moreau, N. Gibbins, D. DeRoure, S. El-Beltagy, W. Hall, G. Hughes, D. Joyce, S. Kim, D. Michaelides, D. Millard, S. Reich, R. Tansley, and M. Weal. SoFAR with DIM Agents: An Agent Framework for Distributed Information Management. In *The Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, pages 369–388, Manchester, UK, Apr. 2000.

[15] C. O'Donovan, M. Martin, A. Gattiker, E. Gasteiger, A. Bairoch, and R. Apweiler. High-quality protein knowledge resource: Swiss-prot and trembl. *Briefings in Bioinformatics*, 3(3):275–284, 2002.

[16] N. Paton, M. Atkinson, V. Dialani, D. Pearson, T. Storey, and P. Watson. Database access and integration services on the grid. In *Fourth Global Grid Forum (GGF 4) Databases and the Grid BOF*, 2002.

[17] R. S. Sandhu and Q. Munawer. How to do discretionary access control using roles. In *ACM Workshop on Role-Based Access Control*, pages 47–54, 1998.

[18] J. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.

[19] Universal Description, Discovery and Integration of Business of the Web. `www.uddi.org`, 2001.

[20] Web Services Description Language (WSDL). `http://www.w3.org/TR/wsdl`, 2001.

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