Principles of Personalisation of Service Discovery

Juri Papay **Simon Miles** Michael Luck Luc Moreau Terry Payne

School of Electronics and Computer Science
University of Southampton
{jp, sm, mml, L.Moreau, trp}@ecs.soton.ac.uk

Abstract

We define personalisation as the set of capabilities that enables a user or an organisation to customise their working environment to suit their specific needs, preferences and circumstances. In the context of service discovery on the Grid, the demand for personalisation comes from individual users, who want their preferences to be taken into account during the search and selection of suitable services. These preferences can express, for example, the reliability of a service, quality of results, functionality, and so on. In this paper, we identify the problems related to personalising service discovery and present our solution: a personalised service registry or *View*. We describe scenarios in which personsalised service discovery would be useful and describe how our technology achieves them.

1 Introduction

Scientists are starting to use distributed services to automate their experiments, and to coordinate them using workflow languages, to take full advantage of the power and flexibility the Grid offers. The pool of available services is constantly being increased, so users cannot know all, or the best, that may be used. In consequence, users, and the software tools they employ, must discover services that will perform the tasks they require by querying *registries* of service adverts. However, public service registries advertise an increasingly large number of services, from which only a small fragment will be relevant for the individual user, project or organisation. Tailoring computational processes to individuals is *personalisation*, an idea that can be applied to service discovery to increase the likelihood of the discovery process resulting in a set of services relevant to the individual, and matching their particular requirements.

Existing service registry frameworks, such as UDDI [5], OWL-S[1] and BioMoby[7], are focussed on providing public service discovery and thus do not address the needs of personalisation. The recently released UDDI version 3 [5] specification moves a small step towards personalisation by allowing clients to subscribe to notification of changes in the content of the registry. By specifying the parameters of subscription the users are informed only about those changes that match the specific parameters. While useful, this does not actually personalise the discovery process, as the same services will be found by the same query to the UDDI registry by any individual. However, despite the numerous limitations, the above standards will continue to evolve and be used by different communities so that, when providing an alternative mechanism for service discovery, existing standards should be *extended* rather than new ones developed.

Personalised service discovery is service discovery in which the personal preferences of the user are taken into account. The most basic way to take into account personal preferences would be to ask the user every time a choice between services was to be made, but this would be both impractical and counter-productive. Our overall objective is to make service discovery easier not just for humans but for the machines as well, by making this process semi-automatic. This involves recording user preferences in a form that is accessible programmatically i.e. in the form of *metadata*. Metadata represents additional information related to, and explicitly associated with, an existing piece of data. In the case of service description, this metadata can, for example, describe the functionality of the service, characterise its reliability, cost, trust rating etc. The process of adding metadata is called annotation, and the availability of metadata is of key importance, as it enables filtering and ordering of the list of services generated by a query according to the user's personal preferences.

In this paper, we present our approach to personalisation of service discovery by describing the design and implementation of a personalised service registry, *View*, developed within the myGrid

project. myGrid [4] is a pilot project funded by the UK e-Science programme [6], with the goal of developing a software infrastructure to provide support for bioinformaticians in the design and execution of *in-silico* experiments utilising the resources of the Grid. The Grid is a truly heterogeneous environment where the resources are geographically distributed and managed by a multitude of institutions and organisations. Discovering services, workflows and data in this fluid and ever-changing environment is a real challenge that highlights the need for registries with reliable information content.

The specific technical contributions detailed in this paper are the following:

1. A set of principles that must be followed to allow service discovery to become personalised.

2. A protocol that enables filtering and replication of the information content of public registries, and stores it in a personalised registry.

3. The design and implementation of a personalised service registry that follows the principles we have established.

The paper is structured as follows. First, in Section 2, we present several scenarios that illustrate the role of personalisation in the bioinformatics context. Section 3 then describes the key features of the *View*, while Section 4 examines in detail the message-passing architecture enabling personalisation. Then, Section 5 discusses the scope of related work, and Section 6 draws conclusions and outlines further work.

2 Personalisation Scenarios

In this section, we describe three scenarios of increasingly complex requirements for personalisation in service discovery, and show how our *View* technology solves this.

2.1 Personal View

In the most basic scenario, a user would like service discovery to be conducted over the most up-to-date adverts for published services but only for those services that are classified as applicable to them. They also want to be able to annotate the service adverts with *opinions* and other forms of extra information that should then be used in the discovery process.

A user can set up a *View* which will pull entries from a set of service registries. For each registry, the user specifies a query to provide the initial data extracted from that source. Changes in the sources that affect the outcome of the queries will cause notifications to be sent to the *View*. By default, any changes are be reflected in the *View* by an automatic update mechanism (which is discussed further below). The user can manually edit the *View* by editing the metadata attached to entries or by deleting entries and, by default, any manually edited data will not be overwritten by the automatic update mechanism.

2.2 Lab View

Moving to the next level of complexity, multiple users in an organisation may want to perform discovery that takes account of the expert information on services that they have collectively built up, but they may want that information to be managed only by reliable organisation members.

Control over the content of a *View* can be shared by a members of an organisation. For example, a biology lab can have a *View* that contains metadata relevant to the members of the lab but with one (or more) designated curator(s) authorised to change the entries and configuration. Then, if a PhD student joins the lab, they are given access to the lab *View* containing the relevant services. In the training period, the student will only be given *read* access to the *View* but, at a later stage, they can have a private *View* created by the curator. Only metadata can be added to the private *View*, but no other modifications are allowed. Later, the *View's* authorisation policy can be changed to allow the student more control, such as modifying metadata and adding/removing services. Eventually, the PhD student graduates and can become the curator of the lab *View*.

2.3 Multiple Views

In the final scenario, different users wish to share experience, and draw on the experience of the most knowledgable organisation members as before, but they wish to perform some filtering of metadata and add extra information that is particular to their use of services.

To achieve this, multiple *Views* can exist and interact. Figure 1 illustrates the scenario in which the expert scientist in an organisation has a personalised registry, View 1, which copies the service adverts published in one or more public registries (Registry 1 to N). The expert then adds a *trust value* as metadata to each service advert, indicating how reliable they have found that service. By contrast, a novice in the same organisation also has a personalised registry, View 2, which copies the content of the expert's registry, but only where the trust value of a service is higher than a particular defined constant. This copying is triggered by a notification from View

1 indicating that an update has occurred. The novice is the only user allowed to edit the metadata in View 2. This means that when the novice discovers services, they are only provided with services that the expert has judged as trustable.

3 Principles of Personalised Service Discovery

In this section, we detail the ideas underlying our personalised service registry. In particular, the following principles were taken into account in designing our middleware.

1. Service discovery should be based on the service adverts currently available in public registries. Service providers should not have to publish services multiple times for users to take advantage of personalised service discovery.

2. While aggregating the contents of public service registries may aid the user in discovering what they need, there are many services that will not be applicable to their requirements, so filtering must take place before, as well as during, discovery.

3. User annotations of service adverts should be taken into account in discovery. However, users may not want their annotations to be publicly accessible, so metadata should be stored locally and access controlled.

The following subsections describe the mechanisms by which the above are achieved in our personalised service registry.

3.1 Metadata attachment

An essential element in personalised service discovery is the ability to augment service descriptions with additional information, i.e. metadata. Service providers may adopt various ways of describing their services, access polices, contract negotiation details, etc. However, many resource consumers also impose their own selection policies on the services they prefer to use, such as quality of service and reputation metrics. A significant contribution of the *View* middleware is also to allow *third parties* to add metadata to service descriptions, so that information about a service can be built up rapidly and used in discovery. Furthermore, it is useful to add such metadata not only to service descriptions, but also to any other concept that may influence the discovery process, such as supported operations, types of arguments, businesses, users.

Such metadata may be structured according to published ontologies, facilitating unambiguous interpretation by multiple users, especially in the case of a public registry; alternatively, such metadata may also be raw and unstructured, in the case of a personal registry used by a single user. The result is an extremely flexible service registry that can be the basis of a sophisticated semantically-enhanced service discovery engine, an essential component of a Semantic Grid.

3.2 Notifications of Change

In order for a personalised service registry to replicate the contents of public registries without the need for re-publishing, there must be a mechanism for determining when a service advert has been added to or updated. Following the approach taken by UDDI version 3 and others, the *View* can send notifications whenever changes to its content take place, which may be useful to a variety of clients. The *View* can also subscribe to receive notifications from other registries, and update its contents according to theirs.

Different kinds of update may occur in a *View* because different parts of an advert can be added, updated or removed. Of these, clients performing discovery will be particularly interested in changes to the service advert as a whole, the metadata annotations to the advert and the service interface definitions (which are registered as WSDL).

To provide the most information for filtering, and to be as useful as possible to other clients, View notifications are classified into *topics*, including the following.

ServiceAdded indicates that a new service has been advertised;

ServiceRemoved indicates that a service advert has been removed from the *View*;

ServiceUpdated indicates that an existing service advert has changed;

MetadaDataAdded indicates that new metadata has been attached to a service description or a business entity;

MetadaDataUpdated indicates that metadata attached to a service description or a business entity has changed;

MetadaDataRemoved indicates that metadata attached to a service description or a business entity has been removed;

WSDLAdded indicates that a new WSDL file has been registered.

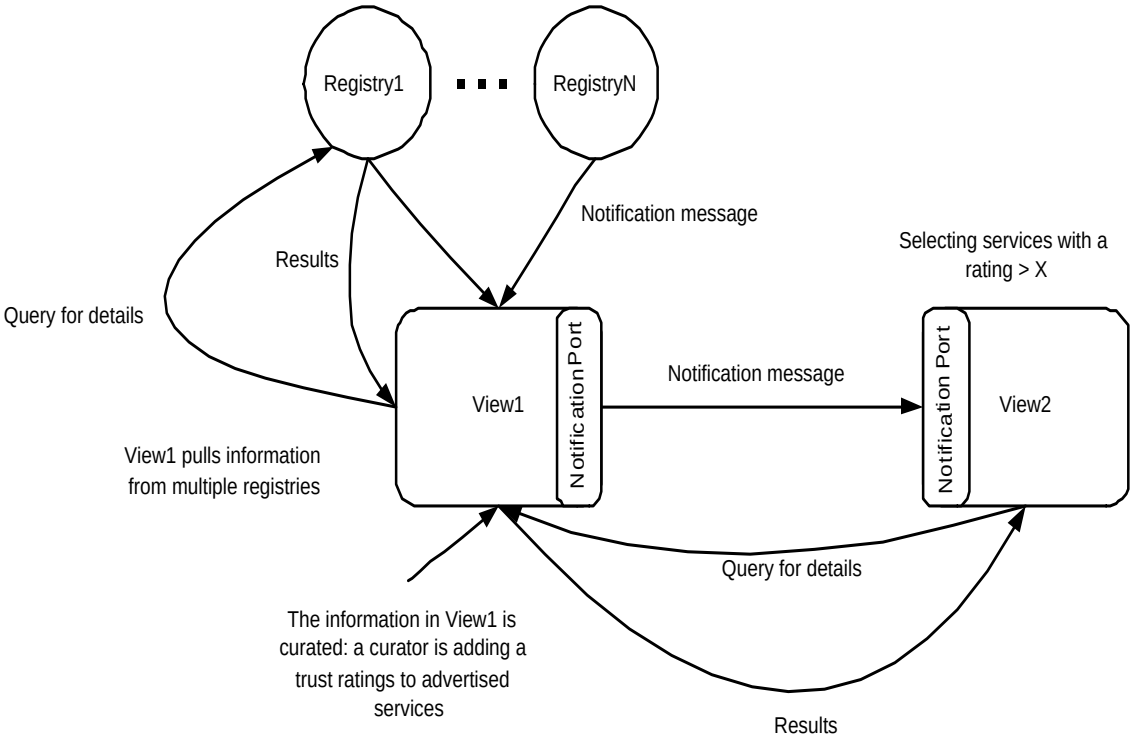

Figure 1: A deployment scenario

WSDLUpdated indicates that a WSDL file advert has been changed or a metadata attached to a part of WSDL file has changed.

WSDLRemoved indicates that a WSDL file advert has been removed.

3.3 Example Interaction

Based on metadata annotation and the provision of notifications, we can see how this applies to the Mutiple Views scenario discussed in Section 2.3 above. The sequence diagram in Figure 2 illustrates the interaction between public registries and *Views* in this scenario. Initially, a service provider publishes a service in one of the public registries, and the public registry sends out a notification message to all subscribers about the new entry. The incoming message is processed by *View 1*, which can query the public registry for service details and, after downloading the information, store it locally. Once the service details have been stored in *View 1*, they can be curated by an expert who can add additional information by attaching metadata. In our example, the expert adds ratings to individual entries. During the startup time, *View 2* registers with *View 1* by subscribing to a particular topic. Upon storing a new entry in *View 1*, each subscriber (in this case there is only one) receives a *NewServiceRegistered* message. On arrival of the message, *View 2* queries *View 1* for more service details and metadata. Then, after obtaining all requested data, *View 2* performs a selection procedure, the outcome of which determines whether to replicate the service description locally or to discard it. In our example, this selection is based on checking the value of the rating and, if the rating is higher than a threshold, then the service description and the associated metadata is stored in *View 2*. Otherwise it is ignored. This simple example illustrates how the content of *View 2* can be personalised according to the value of a single type of metadata.

4 Internal architecture

During the design of *View*, the following software design principles have been taken into account:

Modularity The code is divided into separate modules. In the design of *View*, each module represents a protocol, enabling users to query, update and annotate the content of the personalised registry. At present, the *View* provides supports for the following protocols: UDDI, DAML-S, BioMoby, metadata attachment and WSDL. The modular design enables conceptual seperation, flexible configuration, extensibility and code upgrade.

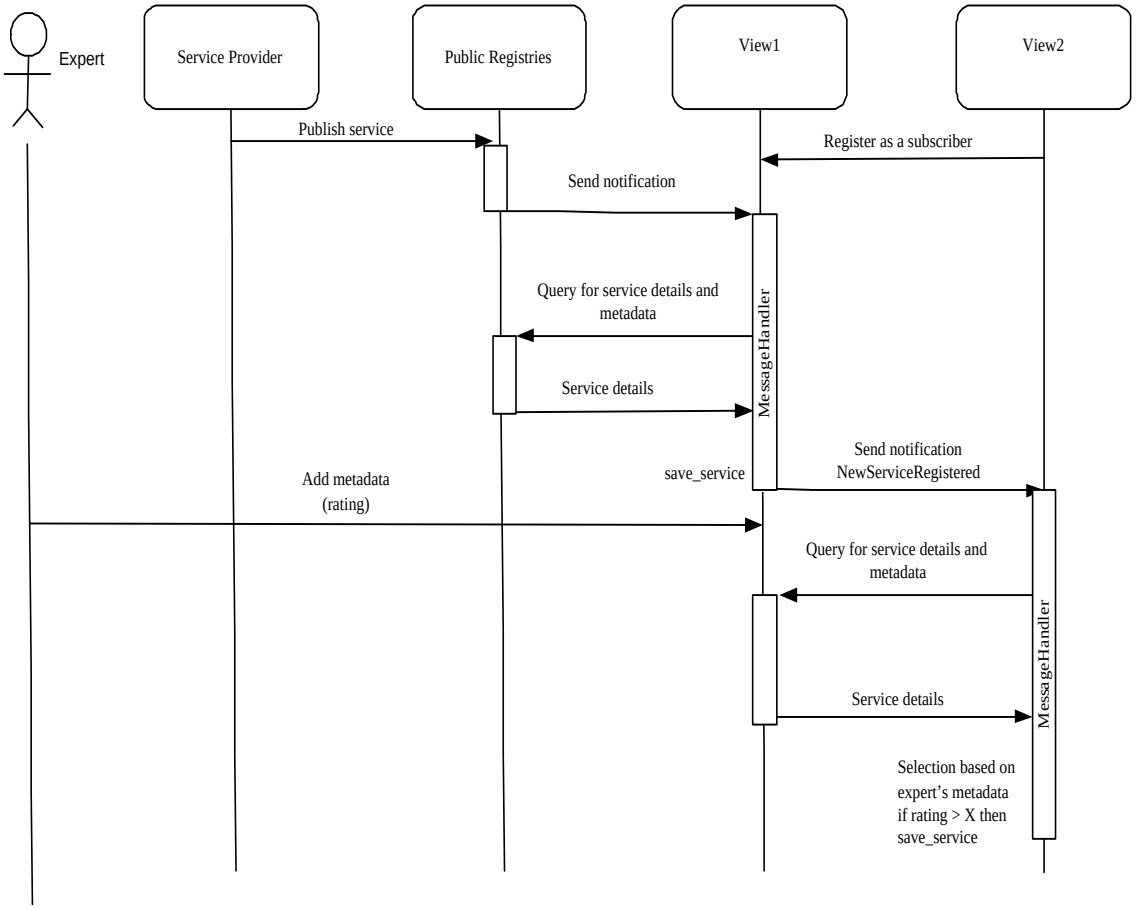

Figure 2: Interaction between multiple *Views*

Generality and Abstraction The whole design is based on interfaces. The API is separated from its implementation, which enables several deployment-time configurations for processing operations in different ways.

Extensibility The *View* should be extensible so that, as new protocols, and new versions of protocols, arise, it can be augmented to be able to process new queries over the existing service advertisements and metadata.

The software is structured in a very modular manner, with the API organised in a series of ports, for publishing and querying, for both service descriptions and metadata. Internally, a message-passing metaphor is adopted, by which messages encode requests and responses of the UDDI and metadata-related APIs. Messages of a given port are handled by so-called "handlers", which either interact with the persistent storage, or produce other messages to be processed by other handlers. These handlers adopt the *visitor* design pattern [3], to ensure static type checking, completeness of the implementation, and consistent error handling. Handlers can be composed in different ways in order to achieve different behaviours (replication, tunnelling, etc.) Thus, multiple configurations of the registry can be achieved at deployment time, by identifying handlers to deal with specific sets of messages. Such modularity allows us to explore alternative implementations that we can test and deploy in separate handlers.

Figure 3 illustrates the APIs, handlers, sequence of method calls, and messages that can occur in a *View*. Each API corresponds to a particular protocol and the implementations of those APIs generate internal messages that can be processed in multiple ways. For example, following the steps originating from the *save_service* method call shown in the figure, the UDDIPublishServer API generates a SaveService message that is processed by the UDDIPublishHandler. The UDDIPublishHandler saves the service advert as per the UDDI business logic, then calls the *newServiceRegistered* method of the RegistryEvent API, which generates the NewServiceRegistered message. In turn, this message is processed by the RegistryEventHandler, which creates a notification message and sends it to the outside world in SOAP format.

The incoming notification is processed by the RegistryEventHandlerIncoming handler of the receiving *View*, which may involve querying the publishing *View* for more details about the service. As a result of this operation, the service details are downloaded from the publishing *View*. After obtaining the service details and metadata, a selection operation is used to determine whether to store the service advert locally or to discard the downloaded information. In this way, the contents of the publising *View* can be fully or partially replicated into a subscribing *View*.

5 Related work

In this section, we examine other service discovery technologies, including service registries, protocols and service capability description languages. For each, we determine how well it can be utilised in personalised service discovery, and describe how our approach differs.

The UDDI service directory (Universal Description, Discovery, and Integration) [5] has become the de-facto standard for service discovery in the Web Services community. Service queries are typically white or yellow pages based: services are located based on a description of their provider or a specific classification (taken from a published taxonomy) of the service type. Service descriptions in UDDI are composed from a limited set of high-level data constructs (Business Entity, Business Service, etc.) that can include other constructs following a rigid schema. While UDDI provides ways to attach a form of metadata to a service advert through the use of tModels, these are neither *personal*, in that they are set by the service providers, nor used in service discovery, as a tModel is a reference to an external technical specification requiring an independent mechanism for reasoning. UDDI version 3 does, however, provide notifications for change, making it ideal as a source of adverts to be personalised in a *View*.

BioMOBY [7] is a service discovery architecture based on a definition of a service as an atomic process or operation that takes a set of inputs and produces a set of outputs. The service, inputs and outputs can all take semantic types; inputs and outputs also have syntactic types. While there is more scope for metadata in BioMOBY than in UDDI in that semantic descriptions can be added for services and used in discovery, it cannot have general extension of service descriptions, such as the attachment of quality of service ratings. As with UDDI, it is centralised and offers no support for personalisation.

The OGSA Registration port type [2] is intended to support Grid Service registration and discovery, with change monitoring via service data query and the OGSA Notification port types. As with UDDI version 3, an OGSA registry is suited to becoming a source for adverts that can be personalised, but does not provide personalised service discovery in itself.

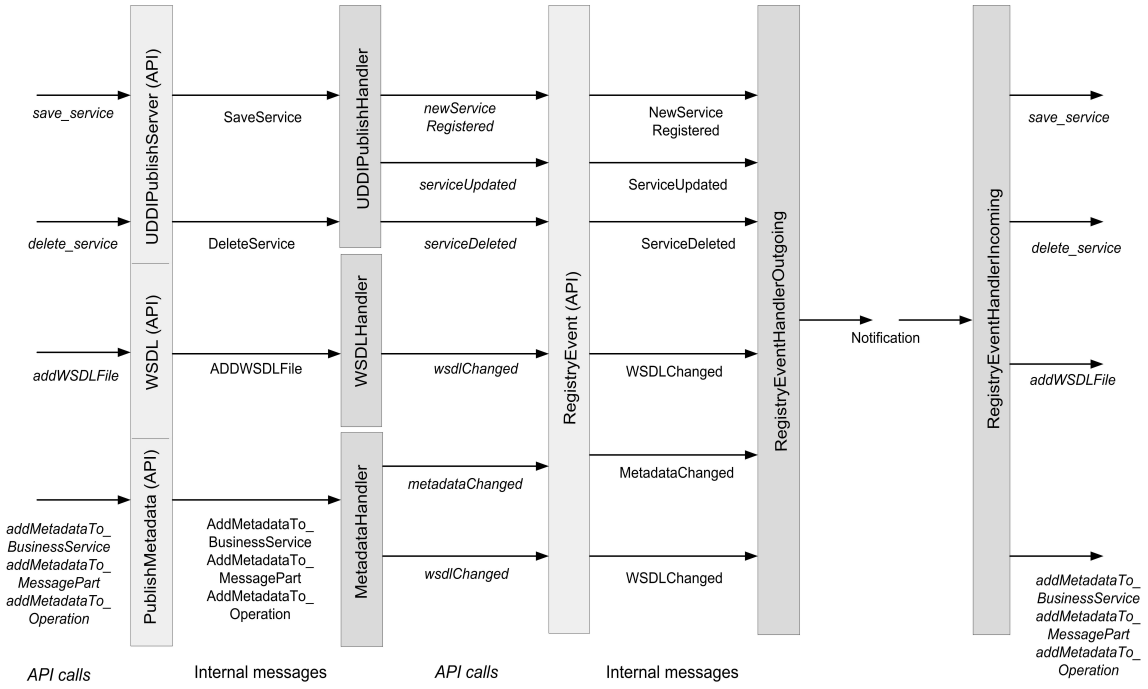

Figure 3: Sequence of messages in a *View*

6 Conclusions and Future Work

In this paper we have investigated the importance of personalisation in the context of service discovery. The problem was investigated in the bioinformatics setting in which service discovery is of key importance, but the results of this research are also applicable to other problem domains.

We have identified the key requirements of personalised service discovery and analysed to what extent these requirements are met by the existing standards. Existing and widely-used service registries and specifications such as UDDI provide little support for personalisation. In response, our research has led to the implementation of a personalised service registry, or *View*, which is a service registry that allows service adverts in other registries to be filtered for applicability to an individual, and then for those adverts to be annotated with extra metadata. These personalised adverts can then be used in discovery, meaning that the results of discovery will be those most useful to the individual.

In developing the *View*, we followed three principles to ensure that users would have personalised service discovery over the most up-to-date services available.

1. Service discovery should be based on the service adverts currently available in public registries. Service providers should not have to publish services multiple times for users to take advantage of personalised service discovery.

2. While aggregating the contents of public service registries may aid the user in discovering what they need, there are many services that will not be applicable to their requirements, so filtering must take place before, as well as during, discovery.

3. User annotations of service adverts should be taken into account in discovery. However, users may not want their annotations to be publicly accessible, so metadata should be stored locally and access controlled.

In future work, we will develop policies that allow those creating and deploying *Views* to more flexibly choose how services from other registries are filtered, and to control access to service adverts and personal metadata in a *View*. We are also testing the *View* rigorously to ensure that it has adequate scalability and performance for deployment in high demand domains.

7 Acknowledgement

This research is funded by EPSRC myGrid project (reference GR/R67743/01). We acknowledge Carole Goble, Phillip Lord and Chris Wroe for their contributions to discussion on the work presented in this paper.

References

[1] DAML-S Coalition:, A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S: Web Service Description for the Semantic Web. In *First International Semantic Web Conference (ISWC) Proceedings*, pages 348–363, 2002.

[2] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The Physiology of the Grid — An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Argonne National Laboratory, 2002.

[3] Erich Gamma, Richard Helm, Ralph Johnson, and Vlissides. *Design Patterns*. Addisson-Wesley, 1995.

[4] myGrid - directly supporting the e-scientist. http://www.mygrid.org.uk/, 2001.

[5] Universal Description, Discovery and Integration of Business of the Web. `www.uddi.org`, 2003.

[6] The UK Research Councils e-Science Core Programme. http://www.research-councils.ac.uk/escience/, 2001.

[7] MD Wilkinson and M. Links. Biomoby: an open-source biological web services proposal. *Briefings In Bioinformatics*, 4(3), 2002.