An Ontology-based Grid Service Discovery Matchmaking Framework

Simone A. Ludwig
Department of Electronic and Computer Engineering, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK

Abstract

The need for semantic matching based on a defined ontology is becoming increasingly important as there are many different implementations available within the Grid Environment (GE). Especially for Service Discovery (SD), an ontology description is very useful in supporting a customized SD process. This paper proposes a SD matchmaking framework for GEs based on a well-defined ontology. The matching of service requests is discussed and a matchmaking mechanism is described in detail. The proposed framework allows for close and customized SD matches that current Service Discovery Systems (SDSs) cannot provide as their SD process is based on type matching only.

Keywords: Ontology; Matchmaking Mechanism; Grid Service Discovery; Service Discovery Matchmaking Framework

1. Introduction

Locating a network service or a device on demand is a challenging task for pervasive computing. A variety of service discovery systems (SDSs) exist that enable an application to discover a service. Most of these systems support an attribute-based discovery as well as a simple name lookup, also called matchmaking, to locate a service. Usually only a set of primitive attribute types, such as string and integer, are used to characterize a service. The Service Discovery (SD) process is therefore primarily done by type matching, based on string or integer comparison. All existing SDSs lack the ability of inexact matching [1]. This is especially the case in Grid Environments (GEs) where so many different implementations of services are available, that might vary in name and functionality. The need for semantic matching becomes increasingly important as more and more services are developed.

Grid computing has been receiving a lot of attention in recent years. A GE typically comprises of heterogeneous resources over a wide-area network. These resources and services are available for Grid applications and therefore, Grid SD is an important issue. Presently, there is effort underway to standardize services. The Open Grid Services Architecture (OGSA) [2] was created to define a base framework of services to achieve interoperability between different Grid implementations. In a GE there is a need to integrate services across distributed, heterogeneous, dynamic "virtual organizations" formed from the disparate resources within a single enterprise and/or from external resource sharing and service provider relationships. This integration can be technically challenging because of the need to achieve various qualities of service when running on top of different native platforms. Building on concepts and technologies from the Grid and Web services communities, this architecture defines uniform exposed service semantics, so called the Grid Service (GS). It defines standard mechanisms for creating, naming and discovering transient GS instances. It provides location transparency and multiple protocol bindings for service instances and supports integration with underlying native platform facilities. OGSA also defines, in terms of Web Services Description Language (WSDL), interfaces and associated conventions, mechanisms required for creating and composing sophisticated distributed systems, including lifetime management, change management and notification. Service bindings can support reliable invocation, authentication, authorization and delegation.

Though, OGSA is defined to provide interoperability and defines discovery of services between different Grid domains, semantic matching based on ontologies will provide a much better SD. It is necessary to capture richer semantics while having more complex data structures. Moreover, a user may not be able to specify the exact values of interested attributes. Thus, approximate matches are desirable.

The term ontology is useful in the GE context, where it describes the need for the provider of a service and the user of that service to share a common understanding of what capabilities the service offers and how they can be put to
use. This is described in more detail in the next section.

This paper is organized as follows. Section 2 gives an introduction to ontology and its development. In section 3, the SD framework including the matchmaking mechanism is presented. Furthermore, the implementation of the ontology for the Grid Job Submission Service written in DAML-S is described. Section 4 presents an account of work related to this subject. Finally, section 5 concludes this paper and gives a perspective to further research.

2. Background to Ontology

When two or more parties seek a common understanding of something, they must work together to ensure that there is a high degree of correlation and similarity between the details of their respective descriptions and definitions of what they are trying to agree on [3]. This implies that shared understanding requires shared definitions. As an example, the day-to-day human interactions are made possible by the fact that our society's members share common understanding and common values. This sharing of common understanding is categorized as the science of ontology, which involves the study of the general concepts and abstractions that make up the fundamental aspects of our world. Over the years, the term "ontology" has grown beyond its original philosophical use and its definition has been blurred slightly when applied to some areas of computing. The term is used in Artificial Intelligence, for example, to describe the fundamental components used to model the worlds that robots understand. Robots can understand only what can be represented as allowed by their ontologies.

Because we pay little attention to the ontology problem in our day-to-day human interactions, we usually take for granted our shared understanding and shared values that allow us to interact with relative ease. We tend not to consider it with respect to computing problems such as computing services.

Particularly, for GEs where SD is a significant issue, the need to share a common ontology becomes very important.

2.1. History of Ontology Development

A prerequisite for widespread use of ontologies is a joint standard for their description and exchange. RDF(S) (Resource Description Framework Schema) itself is an ontology/knowledge representation language which contains classes and properties (binary relations), range and domain constraints (on properties) and subclass and subproperty (subsumption) relations. RDF(S) is a relatively primitive language, however, more expressive power would clearly be necessary and desirable to describe resources in sufficient detail. Moreover, such descriptions should be amenable to automated reasoning if they are to be used effectively by automated processes [4].

These considerations led to the development of the Ontology Inference Layer (OIL) [5] and later to the design of DAML+OIL [6]. DAML+OIL is a more recent proposal for an ontology representation language that has emerged from work under DARPA's Agent Markup Language (DAML) initiative along with input from leading members of the OIL consortium. DAML+OIL is based on the original OIL language, but differs in a number of ways. DAML+OIL provide a greater interoperability on the semantic level. In this way, DAML+OIL extends the RDF(S) basic primitives for providing a more expressive ontology modeling language and some simple terms for creating inferences. In particular, DAML+OIL has moved away from the original frame-like ideas of OIL. It is an alternative syntax for a Description Logic (DL).

As part of the DARPA Agent Markup Language program, an ontology of services, called DAML-S [7] has been developed, which provides a set of basic concepts and relations for declaring and describing services, by utilizing the ontology structuring mechanisms provided by DAML.

The DAML-S service is characterized in three types which are service profile, service model and service grounding [8]. This is shown in Figure 1. The service profile describes what the service does. It provides the type of information needed by a service requester to determine whether the service has the desired capabilities. The service model, also named process model, describes how the service works, i.e. how it is composed and what happens when the service is executed. A service grounding specifies the details of how a service can be accessed. The profile provides the information needed for the discovery of a service. The service model and grounding describe how service requesters and providers can access and interoperate with each other.

Service profiles consist of three types of information:
1. A human readable description of the service.
2. A specification of the functionalities that are provided by the service.
3. A list of functional attributes that provide additional information and requirements about the service that assist when reasoning about several services with similar capabilities.

Table 1
<table>
<thead>
<tr>
<th>Description Properties of the Service Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
</tr>
<tr>
<td>intendedPurpose</td>
</tr>
<tr>
<td>textDescription</td>
</tr>
<tr>
<td>role</td>
</tr>
<tr>
<td>requestedBy</td>
</tr>
<tr>
<td>providedBy</td>
</tr>
</tbody>
</table>

Table 2
<table>
<thead>
<tr>
<th>Functional Attributes of the Service Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>geographicRadius</td>
</tr>
<tr>
<td>degreeOfQuality</td>
</tr>
<tr>
<td>serviceParameter</td>
</tr>
<tr>
<td>communicationThru</td>
</tr>
<tr>
<td>serviceType</td>
</tr>
<tr>
<td>serviceCategory</td>
</tr>
<tr>
<td>qualityGuarantees</td>
</tr>
<tr>
<td>qualityRating</td>
</tr>
</tbody>
</table>

Service functionalities are represented as a transformation from the inputs required to the outputs produced. For example, a news reporting service would advertise itself as a service that, given a date, will return the news reported on that date. Functional attributes specify additional information about the service, such as what guarantees response time or what accuracy it provides or the cost of the service. Table 1 and 2 list the properties defined by the service profile as described above.

3. Service Discovery Framework

Computational Grids, emerging as an infrastructure for the next generation computing, enable the sharing, selection and aggregation of geographically distributed heterogeneous resources for solving large-scale problems in science, engineering and commerce. As the resources in the Grid are heterogeneous and geographically distributed with varying availability and variety of usage and cost policies for diverse users at different times, priorities as well as goals of both users and owners vary with time. The management of resources and services in such a large distributed environment is a complex task. Therefore, SD plays an important role in such environments.

The following is a description of a proposed SD framework for GE's. The framework relies on an ontology description, that allows semantic matching and it is based on the concepts of the LARKS matchmaker proposed by Sycara et al. [9]. The matching mechanism comprises of three filter stages. These are context, syntactic and semantic matching whereas the service ontology database provides the knowledge-base.

The difference between the LARKS and the Grid SD matchmaker is that the Grid SD relies on DAML-S and its ontologies for the matchmaking, while LARKS only uses the set of filters that progressively restrict the number of advertisements which are candidates for a match. The Grid SD achieves semantic matching as it relies on the set of filters and also allows the flexibility of close matches by providing an ontology knowledge-base.

The description of the matching process in the SD matching framework [10] is described below, and is followed by a description of the illustrated framework.

3.1. Matching Mechanism

An advertisement matches a request, when the advertisement describes a service that is sufficiently similar to the service requested [11]. The problem of this definition is to specify what “sufficiently similar” means. Basically, it means that an advertisement and a request are “sufficiently similar” when they describe exactly the same service. This definition is too restrictive, because providers and requesters have no prior agreement on how a service is represented and additionally, they have very different objectives. A restrictive criterion on matching is therefore
bound to fail to recognize similarities between advertisements and requests.

It is necessary to allow matching engines to perform flexible matches, those that recognize the degree of similarity between advertisements and requests in order to provide a softer definition of "sufficiently similar". Service requesters should be allowed to decide the degree of flexibility that they grant to the system. If they allow little flexibility, they reduce the likelihood of finding services that match their requirements, which means they minimize the false positives, while increasing the false negatives. On the other hand, by increasing the flexibility of a match, they achieve the opposite effect, that is, they reduce the false negatives at the expense of an increase of false positives.

An additional problem related with performing flexible matches is that the matching engine is open to exploitation from advertisements and requests that are too generic in the attempt to maximize the likelihood of matching. For instance, a service may advertise itself as a provider of everything, rather than being precise with what it does, and similarly, the requester may ask for any service, rather than specifying exactly what it expects. The matching engine can reduce the efficiency of these exploitations by ranking advertisements based on the degree of a match supplied with the request.

Considering all these criteria, the matching engine should satisfy the following [11]:

a) The matching engine should support flexible semantic matching between advertisements and requests based on the ontology available to the service and the matching engine.

b) Despite the flexibility of the match, the matching engine should minimize false positives and false negatives. Furthermore, the requesting service should have some control on the amount of matching flexibility it allows to the system.

c) The matching engine should encourage providers and requesters to be precise with their descriptions at the cost of either not being matched or being matched inappropriately.

d) The matching process should be efficient which means that it should not burden the requester with excessive delays that would prevent its effectiveness.

The algorithm proposed aims to satisfy all four requirements. Semantic matching is based on DAML ontologies. The advertisements and requests refer to DAML concepts and the associated semantic. By using DAML, the matching process can perform implications on the subsumption hierarchy leading to the recognition of semantic matches despite their syntactical differences and difference in modeling abstractions between advertisements and requests. The use of DAML also supports accuracy, which means that no matching is recognized when the relation between the advertisement and the request does not derive from the DAML ontologies used by the registry. Furthermore, the semantics of DAML-S descriptions allow the definition of a ranking function which distinguishes multiple degrees of matching.

### 3.2. Service Discovery Matchmaker

Three components are necessary for the Grid SD matchmaking. These are service provider, service requester and service discovery matchmaker. Figure 2 shows the SD framework, where the sequence of interactions is as follows:

1. The service provider (GS) registers its service description in the service registry database.
2. The Grid application requests a GS and sends the request to the SD matchmaker.
3. The matchmaker returns the matches to the service requester.
4. The service requester decides then which service to use depending on the client’s need and contacts the service directly.

GSs are services that are provided in a GE. These services are, for example, authentication, authorization, job submission, distributed job scheduling, resource optimization, data management, wide-area data transfer, file replication, resource management and resource monitoring. GSs can be defined as a bundle or a collection of “simple” services. These are more complex and are registered in the service registry database.

The service requester consumes services offered by GS providers in the system. A request for any GS has to be sent to the matchmaker. In a GS, service requesters are the applications which use the GSs. These are, for example, the LHC-HEP (Large Hadron Collider - High Energy Physics).
experiments such as Alice [12], Atlas [13], CMS [14] and CDF/D0 [15]. These LHC-HEP experiments want to access the GSs in order to process their petabytes of data necessary for their experimental evaluations.

The service matchmaker mediates between service requesters and service providers for some mutually beneficial cooperation. Each provider must first register with a registry, also called matchmaker. Service provider advertises their capabilities (advertisements) by sending some appropriate messages describing the kind of service they offer. Every request a matchmaker receives will be matched with his actual set of advertisements. If the match is successful, the matchmaker returns a ranked set of appropriate service providers and the relevant advertisements to the requester.

In contrast to a broker, a matchmaker does not deal with the task of contacting the relevant providers, transmitting the service request to the service provider and communicating the results to the requester. This avoids data transmission bottlenecks, but it might increase the amount of interactions between service requesters and matchmaker.

The matchmaker processes a received request in the following three basic steps:

a) Comparison of the request with all advertisements in the service registry database.

b) Decision of the service provider whose capabilities match best with the request depending on the specified algorithm and the defined service ontology. Every pair of request and advertisement has to go through several different filters during the matchmaking process.

c) Providing information to the service requester by sending a contact address and related capability descriptions of the relevant service provider.

The ontology of a matchmaker is not equal to the union of local domain ontologies of all service providers who are actually registered at the matchmaker. This also shows an advertisement database is important for the first filter stage.

Thus, a matchmaker has only partial knowledge and might not be up-to-date with the actual time of processing incoming requests. This is due to the fact that, for efficiency reasons, changes in the local ontology of a service provider might not be propagated immediately to all matchmaker it is registered at. These are the implications of a centralized SD model for the Grid SD matchmaker. A better solution is a decentralized SD model following the distributed nature of the Grid. In distributed systems components may fail, messages may be lost or services expire, hence a management for the lifetime of services must be provided to allow a secure SD process.

The architecture fulfills the matching criteria listed in section 3.1 as follows. The Grid SD matchmaker supports flexible semantic matching between advertisements and requests based on the ontology available. Minimizing false positives and false negatives is achieved with three filter stages in combination with the well-defined ontology. The filter stages restrict the false positives and the ontology restricts the false negatives. The matching engine should encourage providers and requesters to be precise with their descriptions. To achieve this, the service provider follows an XML-based description to advertise its services and the service requester generates a query in a specified format. Defining the ontology and the filter stages precisely allows the matching process to be efficient.

3.3. Component Description of the Matchmaker

When a Service Requester requests a service the request goes through the parser. The parser breaks data into smaller elements, according to a set of rules that describe its structure. Most data can be decomposed to some degree.

The matching process of the matchmaker is designed with respect to the criteria listed in the first section. The matching process is organized as a series of increasingly stringent filters. This means that matching a given request into a set of advertisements consists of the following five filters that are organized in three consecutive filtering stages:

1) **Context Matching** - Selects those advertisements in the auxiliary database that can be compared with the request in the same or similar context.

2) **Syntactic Matching** - This filter compares the request with any advertisement selected by the context matching in three steps that are comparison of profiles, similarity matching and signature matching. The request and advertisement profile comparison uses a weighted key-word representation for the specifications and a given term frequency based similarity measure. The service ontology provides the service model and the service grounding. The last two steps focus on the input/output constraints and declaration parts of the specification.

3) **Semantic Matching** - This final filter checks if the input/output constraints of any pair of request and advertisement logically match.

For reasons of efficiency, the context filter roughly prunes off advertisements that are not relevant for a given request.

In the following two filtering stages (syntactic and semantic matching) the remaining advertisements in the auxiliary database of the matchmaker are checked in more detail. All filters are independent from each other. Each of the filters narrows the set of matching candidates with respect to a given filter criteria.

The service registry database contains all GSs as the DAML-S service profile structure. The auxiliary database comprises a database for word pairs and word distances, basic type hierarchy and internal data. The service ontology database contains the DAML-S service ontology that was
described in section 2. The service registry database is linked to the service ontology database and both contain the DAML-S service ontology. The main benefits of providing a local ontology are as follows. The user can specify in more detail what is being requested or advertised, and particularly the matchmaker is able to make automated inferences on these additional semantic descriptions. This improves the overall quality of the matching process. There are two major types of ontologies defined, the domain and the task ontology. Both ontologies are incorporated in the service ontology database. The domain ontology provides the environment with the knowledge of what it knows. The task ontology provides the environment with the knowledge of what it is doing.

In front of the ontology database is the ontology reasoner. The reasoner, offered by a Description Logic (DL), supports the development and incremental maintenance of an ontology. Highly optimized implementations of sound and complete subsumption algorithms for expressive DLs can be used in spite of the complexity. Thus, ontologies expressed using DAML+OIL can be verified by using e.g. the DaCT reasoner. The key reasoning services of the DaCT reasoner are subsumption checking, classification and concept satisfiability. Subsumption checking is done between two concept descriptions. Classification organizes a collection of concept expressions into a partial order based on the subsumption check. This provides a lattice of definitions, ranging from the general to the specific. Composed definitions have their position implicitly determined. Classification is a dynamic process where new compositional expressions can be added to an existing hierarchy. Concept satisfiability checks whether a concept description can never have instances because of inconsistencies or contradictions in the model.

3.4. Implementation of the Ontology written in DAML-S

The Grid Job Submission Service (JSS) is part of the Grid ontology written in DAML-S\(^1\) and is described in detail as follows. JSS is responsible for the actual job management operations (submission, cancellation and monitoring) once the Resource Broker (RB) has chosen a suitable Computing Element for running the job. The RB is a middleware that supplies distributed clients with job execution in a heterogeneous computing environment such as the Grid. Client applications are provided with a set of interfaces for sending requests and receiving responses to and from the RB. The RB is responsible for carrying out tasks to satisfy the client requests. These tasks include e.g. interacting with the Replica Catalog and performing the job submission via the JSS.

The JSS comprises the following steps:
1. Check certificate validity.
2. Create JDL (Job Description Language) for the job.
3. Submit the job using a net process or any more complex procedure.
4. Monitor the load via network.
5. Verify correct job execution and progress.
6. Periodically check the log files such as stdout and stderr.
7. Verify correct job termination and output.
8. Update bookkeeping of processed data.

The code fragments in the appendix describe the implementation of the service, profile and process ontology of the Grid JSS. Figure 3 shows the service model which describes the job submission service and refers to the service profile and the service process. The service profile, shown in Figure 4, defines the specification of the functionalities. These are provided by the service and a list of functional attributes which offer additional information and requirements about the service. Figure 5 shows the process model. It describes how the service works, i.e. how the service is composed and what happens when it is executed. There are atomic, simple and composite processes which can be modeled with DAML-S. The job submission is an atomic process identified by the \&_process attribute.

In this section the SD matchmaking framework based on ontology knowledge was described. It revealed how the matchmaking process is done and how close and customized matching of services is achieved. The implementation of the ontology described for the Grid JSS concluded this section.

4. Related Work

During the past few years lots of effort and research have been placed in the field of knowledge representation and semantic matching mechanisms which are described in the following paragraphs. The different approaches are based on resource sharing and management, description logics for knowledge representation, enhanced service discovery methods and the usage of ontology-based knowledge representation.

4.1. Knowledge Grid Platform VEGA-KG

VEGA-KG (Knowledge Grid Platform) allows a uniform sharing and managing of knowledge resources across the Internet [16,17]. It includes two major components: a resource space model (RSM) and an operable knowledge browser. The RSM uniformly organizes information, knowledge and service resources and provides a local and a universal resource view for resource operations. The knowledge browser enables users to locate and manage resources conveniently by using a set of operations. The
service discovery process is done by the knowledge browser in the following steps. The browser locates the knowledge resources in the knowledge space via the
coordinates first. It then selects suitable operations and parameters, and delivers the operation(s) to the execution engine. It receives and eventually shows the operation results.

4.2. Clustering of Soft-Devices
Soft-Devices without a doubt are promising next-generation web resources [18]. These are software mechanisms which provide services to each other and to other virtual roles regarding the content of their resources.

Soft-Devices can play both the producer and the consumer role.

The soft-device consists of six major components:
- A container stores the content in a machine-compatible manner.
- A detector which is responsible for detecting the requirements.
- An explainer explains the resources content.
- Multiple built-in workflows enable the soft-device to work according to different requirements.
- A knowledge base supports the explainer, detector and adaptive workflows.
- An interface supports the producer in defining resources’ content.

The advantage of the soft-device sharing compared to the
Web-based information sharing is that for the soft-devices, machine-compatible semantics information can be included. The service discovery process works by the consumers posting their requirements on the published requirements list first. After that, the soft-devices actively search the list and inform the broker if the requirements match their capabilities. The broker then selects the best soft-device, clusters particular soft-devices to provide an integrated service or adapts existing clusters to respond to changes in consumers’ requirements.

4.3. Description Logics Matchmaker

DLs are a family of knowledge representation formalism. They are based on the notion of concepts and roles, and are mainly characterized by constructors that allow complex concepts and roles to be built from atomic DLs [19]. The main benefit from these knowledge languages is that sound and complete algorithms for the subsumption and satisfiability problems often exist. A DL reasoner solves the problems of equivalence, satisfiability and subsumption. The matchmaking service provides three basic functionalities which are advertising, querying and browsing. The algorithm is a translation in DL terms of the ideas exposed, previously mentioned. The DL Matchmaker achieves SD matching based on knowledge representation formalism.

4.4. DReggie

The project DReggie is an attempt to enhance the matching mechanisms in Jini and other SDSs. The key idea in DReggie is to enable these SDSs to perform matching based on semantic information associated with the services [20]. Semantic service matching introduces the possibilities of fuzziness and inexactness of the response to a SD request. In the DReggie system, a SD request contains the description of an “ideal” service - one whose capabilities match exactly with the requirements. Thus, matching now involves comparison of requirements specified with the capabilities of existing services. Depending on the requirements, a match may occur even if one or more capabilities do not match exactly. Service description in DReggie system is marked up in DAML. The semantic matching process, which uses these descriptions, is performed by a reasoning engine. At the heart of DReggie is an enhanced Jini Lookup Service (JLS) that enables smart discovery of Jini-enabled services. DReggie retains the matching mechanism currently employed by the Jini lookup and discovery infrastructure.

4.5. Bluetooth Service Discovery Protocol (BSDP)

BSDP supports a sophisticated matching mechanism that uses semantic information associated with services and attributes to decide the success or failure of a query. The enhanced version of BSDP (presented by Saikanth Avanch et al. [21]) support semantic matching and provide service registration. The first version used the RDF/RDF-S data model and the second and current version uses DAML.

4.6. Semantic Search - SHOE

SHOE (Simple HTML Ontology Extensions) is an ontology-based knowledge representation language designed for the Web. The current version of the language is the result of an effort that started in 1996 and anticipated many of the features which were subsequently added to XML and RDF. SHOE uses knowledge oriented elements, and associates meaning with content by making each web page commit to one or more ontologies [22]. SHOE ontologies permit the discovery of implicit knowledge through the use of taxonomies and inference rules, allowing content providers to encode only the necessary information on their web pages, and to use the level of details that is appropriate to the context.

4.7. Process Matching

Matching between two workflow processes is the key step of workflow process reuse. The matching degree between two workflow processes is determined by the matching degrees of their corresponding sub-processes or activities. Furthermore, the matching degree between two
activities is determined by the activity-distance in the activity-ontology repository. This approach of process matching can increase the flexibility of the matching process by using the multi-valued process specialization relationship between the well-defined workflow processes which is described in [23].

5. Conclusion

In the GE where so many different implementations are available the need for semantic matching based on a defined ontology becomes increasingly important in order to provide close and customized service request matches. This paper proposed a new semantic SD framework for GE. It showed the new matchmaking mechanism based on ontology knowledge. The Grid ontology for the JSS (written in DAML-S code) was presented in order to show how ontologies are defined and implemented. The proposed matchmaking framework allows for a better SD and close matches in a flexible way based on the defined ontology. Therefore, Grid applications are able to specify the criteria a service request should be matched with. A link to other semantic and ontology representations such as the Knowledge Grid Platform (VEGA-KG), Soft-Devices, DLs, DReggie, BSDP, SHOE and Process Matching were accounted for.

Further work includes the improvement from a centralized to a distributed SD model as previously mentioned. A solution could be a decentralized model for SD, e.g. like the Web Services Inspection Language (WS-Inspection) used for Web services [24]. Service descriptions in WS-Inspection may be stored at any location, and requests to retrieve the information are generally made directly to the entities which are offering the services. The WS-Inspection specification relies upon existing Web technologies and infrastructure to provide mechanisms for publishing and retrieving its documents. This would serve the distributed nature of the Grid and involves the management of the lifetime of services which provide a secure SD process. Nevertheless, the security of such a system needs to be investigated.

Acknowledgment

The author would like to thank Dr. M. Reyhani for his helpful discussions and useful suggestions.

Part of this research is funded by PPARC (Particle Physics and Astronomy Research Council, UK) for the IST Program of the European Union (Grant IST-2001-32459).

Appendix

Fig. 3. Code Fragment of JobSubmission-Service.daml

Fig. 3. Code Fragment of JobSubmission-Profile.daml
This service provides the job submission service for the Grid environment.

<!-- Descriptions of the parameters -->
<profile:input>
  <profile:ParameterDescription rdf:ID="CreateJDL">
    <profile:parameterName> CreateJDL </profile:parameterName>
    <profile:restrictedTo rdf:resource="&concepts;#JDL"/>
    <profile:refersTo rdf:resource="&js_process;#createJobDescriptionLanguage_In"/>
  </profile:ParameterDescription>
</profile:input>
...
<profile:input>
  <profile:ParameterDescription rdf:ID="UpdateBookkeeping">
    <profile:parameterName> UpdateBookkeeping </profile:parameterName>
    <profile:restrictedTo rdf:resource="&concepts;#Bookkeeping"/>
    <profile:refersTo rdf:resource="&js_process;#updateBookkeeping_In"/>
  </profile:ParameterDescription>
</profile:input>

<!-- The consequence of the job submission is that the certificate is valid -->
<profile:effect>
  <profile:ParameterDescription rdf:ID="ValidCertificate">
    <profile:parameterName> ValidCertificate </profile:parameterName>
    <profile:restrictedTo rdf:resource="&concepts;#HaveValidCertificate"/>
    <profile:refersTo rdf:resource="&js_process;#HaveValidCertificate"/>
  </profile:ParameterDescription>
</profile:effect>

Fig. 4. Code Fragment of JobSubmission-Profile.daml
Fig. 5. Code Fragment of JobSubmission-Process.daml

References