# An Autonomic Service Discovery Mechanism to Support Pervasive Device Accessing Semantic Grid

**Tao Guan*, Ed Zaluska, David De Roure**
School of Electronics and Computer Science
University of Southampton
Southampton, UK
{tg04r, ejz, dder}@ecs.soton.ac.uk
*Corresponding author

**Abstract:** One of the essential challenges of integrating pervasive devices into Grid environment to enhance pervasive device capabilities is that pervasive devices need to locate, find, select and invoke the appropriate Grid services in an autonomic and flexible way. However, at this stage, both Grid service description and discovery standards are still at an immature stage. This paper presents the research work of building a Grid service discovery middleware with Semantic Web technologies. Semantic Web technologies benefit the demand of an efficient discovery of various Grid services for pervasive devices by adding the machine-processable explicit knowledge into the interaction between pervasive devices and Grid services. The middleware has been implemented and interact correctly with other service-oriented pervasive Grid middleware, providing an enhanced Grid access for pervasive devices.

**Keywords:** Pervasive Devices; Grid Services; Semantic Service Matching.

## 1 INTRODUCTION

Grid computing enables versatile resources around the world to be effectively shared, used and managed, instead of limiting users to local equipments. As a service-oriented approach is increasingly adopted, many systems which can discover Grid resources on-the-fly and access them dynamically come into existence. At the same time, two significant trends are in the area of ubiquitous computing with rapid improvements of the modern computing technology: more pervasive devices are deployed; more integration and computation power is required. Although new generation pervasive devices are gradually improving their absolute capabilities (e.g. mobile phones, PDAs), it is still a challenging objective to create complicated applications on them because pervasive devices are typically resource-constrained, relative to their static counterparts (e.g. desktops, workstation).

One feasible solution is that pervasive devices make use of Grid services so that pervasive users are able to access distributed resources automatically on demand with appropriate quality of service delivery. Various Grid services enhance capabilities of pervasive devices and complicated tasks can be completed through user handheld devices. However, the combination of Grid and pervasive computing models enables us to step into a new realm of high-performance Grid access through resource-limited pervasive devices, and it is quite evident that a great number of challenges are required to be solved before the vision of building the bridge between Grid and pervasive computing is realized. The detailed challenges are discussed in (Roure et al. 2006). In the previous paper (Guan et al. 2006), we discussed a system architecture which integrates pervasive devices into the service-oriented Grid environment in an open and flexible way. Another important challenge is that at present, Grid service description and discovery mechanisms are still at an immature stage. Semantic web technologies, providing a very considerable degree of automatic processing, interoperation and integration, will help us to implement an effective Grid service matching mechanism.

With the proliferation of Grid services, semantic specifications of Grid services are gradually becoming a necessary requirement of the automatic, flexible service provision and utilization necessary for Grid clients to perform various tasks. Basically, it is not easy for a request to locate required services in order to execute a task in a service-rich Grid environment. Semantics of Grid services abstract top-level concepts and relationships between concepts so that both service discovery and automatic conversion of interaction formats between heterogeneous services can be realized. Furthermore, a semantic definition mechanism provides a comprehensive representation of a variety of Grid service aspects, building an essential foundation for possible automatic behaviors throughout the whole Grid service development lifecycle.

In this paper, we have presented a service matching mechanisms by using Semantic Web technologies to support pervasive devices accessing Grid services. The rest of this paper is organized as follows. Section 2 introduces the related work and section 3 presents the system architecture briefly. Section 4, 5, and 6 discuss the Grid service matching methodology, the service description and the service discovery algorithm detailedly, with its implementation and experimental results in section 7. Section 8 concludes our current research work and discusses further directions.

## 2 Related Work

Generally speaking, there are two possible roles for pervasive devices in the Grid environment. The first role is that pervasive devices are considered to be a physical interface to the Grid. In this role, pervasive devices act as the Grid service consumer, enabling users to initialize a task on their portable devices, submit a job request, monitor the job execution, and collect results from the Grid to achieve their tasks. The second one is that pervasive devices act as Grid nodes, which are Grid resource providers rather than service consumers. No matter which role that pervasive device is, the current mature Grid infrastructure and architecture do not take into account the design of supporting pervasive systems because pervasive devices had always been considered not to be well suitable as a Grid computing interface and resource node. However, in the past several years, many research projects and papers have been investigating the possibility of integrating pervasive devices into the Grid environment (Millard et al. 2005) (Bhagyavati & Kurkovsky 2003) (Kurkovsky et al. 2004).

One of the challenges required to be overcome to implement the vision of extending pervasive devices into Grid services is to provide a means for advertising resources availability to enable pervasive devices to locate and find the appropriate Grid services. Service discovery protocols simplify the interaction between service consumers and service providers. Various existing service discovery protocols have been introduced during the past several years. In the field of pervasive computing, the example description and discovery solutions are Bluetooth Service Discovery Protocol (BluetoothSIG 2001), Jini (SunMicrosystems 2001), and Universal Plug and Play (UPnP) (MicrosoftCorporation 2000). In the field of Web Services, Universal Description Discovery and Integration (UDDI) (OASIS 2005) is a platform-independent and XML-based registry which enables businesses to publish service listings and discover each other. However, all of these service discovery mechanisms do not support flexible matching between service advertisements and requests, and users cannot locate services automatically on the basis of the capabilities services provide.

Semantic Web technologies give web services rich semantic specification to enable flexible automation of service provision and use. A number of semantic-based web service matching mechanisms have been developed by using Semantic Web technologies, such as (Srinivasan et al. 2006) (Majithia et al. 2004) (Li & Horrock 2003) and so on. They build a semantic layer between users and the web service WSDL description, which makes it possible to compose the web service request with high-level abstract concepts rather than syntactic level terms, addressing several limitations in traditional web service discovery techniques. However, most of these work performs service matching with service function attributes (e.g. service inputs and outputs), and their purpose is the service discovery in the enterprise environment. Our system architecture, on the other hand, combines both Grid and pervasive computing. Hence, other service attributes except service inputs and outputs are required to be considered when designing a semantic service matching middleware for the service-oriented pervasive Grid environment, for example, service contexts and service types.

## 3 System Architecture

In our system architecture, pervasive devices access Grid services through their "deputy" objects, which is created by the middleware in the device initialization stage. As a program that interacts with distributed resources on behalf of pervasive devices, the deputy object is responsible for analyzing tasks from pervasive devices, preparing detailed access and executing mechanism, collecting and storing task results from Grid services.Figure 1 shows a diagram of the overall system architecture. The system architecture attempts to overcome both the slow processing capability of pervasive devices and the data transmission through unreliable wireless network with the limit bandwidth.

The static distributed resources and pervasive devices are interconnected by the Grid gateway. The Grid gateway is a small server available for nearby pervasive devices within the covered range through the local wireless network, providing a relatively resource-rich, stable execution environment and network connectivity when compared to handheld devices. Also note that three kinds of middleware exist in the Grid gateway: the mobile deputy middleware performs a reliable task management mechanism, including task submission, execution, monitoring, result storing on behalf of pervasive users, by accepting and packing tasks as the deputy object; the service-based Grid middleware provides access interfaces for Grid service invocation to

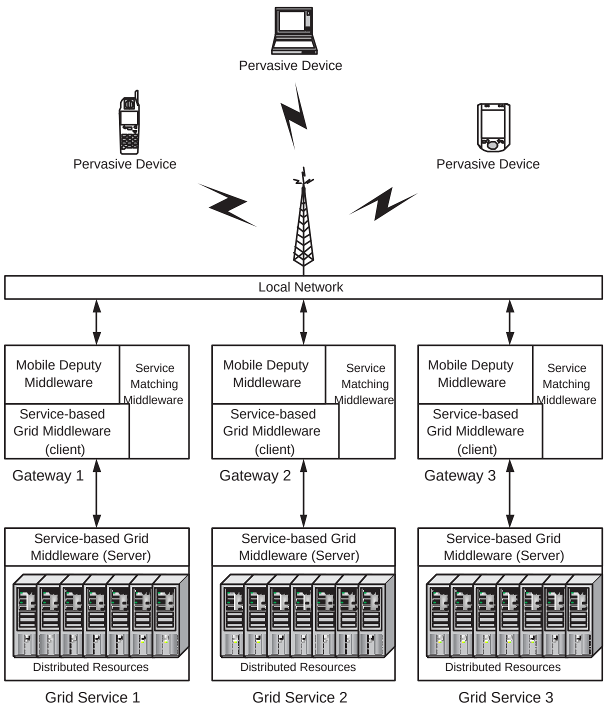

Figure 1: System Architecture.

achieve users' task; the service matching middleware is responsible for the implementation of Grid service registration, discovery, composition and management for pervasive user locating and finding required Grid services to perform their tasks.

## 4 Methodology for Grid Service Discovery

A semantic approach for service matching requires a service description with the ontology definition, and a service search engine with reasoning mechanisms. Ontology is the specification of a conceptualization of a knowledge domain. It is a group of controlled vocabularies that describe objects and their relationships to express something meaningful within a specific interest field. An ontology approach is usually used to present services with abstract and high-level concepts for the service description and a logical reasoning mechanism is usually used to build a service search engine. As long as users describe their service requirements with terms from the same ontology model used to build the service descriptions, logical reasoning mechanisms can find the semantic similarity between the service description and the user requirements, and the matching services will be discovered and returned to users.

The service provider represents all characteristics of a service in the service description. Here, we define a term, *description collection*, which indicates all possible attributes to be described for a service. The attribute may be either a concept or a restriction for existent concepts. For each individual service attribute, an ontology is usually designed to illustrate the definition of the attribute and its relationship with other service attributes. The Grid service description can be completely separated from the detailed Grid service implementation. After Grid service providers deploy their services on the Grid computing platform, they need to publish the service description information in a service registry. Different service descriptions can be made for one service, which enables a service to be reused for several purposes.

Similar to the service description, a *service request* often consists of many individual requirements, specifying the service attribute to be expected in a service. These requirements may include service outputs, effects, inputs, function, location or any possible attributes in terms of the different service request. For a specific service request, all of the requirements can be divided into two categories, a group of strict requirements and a set of general requirement. The strict requirement indicates that this kind of requirement is essential for the service request and has to be met precisely in the service matching, while the general requirement means this kind of requirement is not as important as the strict one and only a rough matching is necessary between the requirement and the related service attribute.

Although we assume that the service request attempts to describe expected requirements with terms from the same ontology model used to build the service description, it is impractical that every service request will get the exact same service even though the required services have already been deployed and advertised because one service could have a number of description formats. In fact, the responsibility of the service search engine is to obtain all of the related services including those that deviate from the request in certain degrees. These deviation services should not be discarded but be classified in an predefined rule (e.g. matching degree). The service request determines which service will be selected based on the information returned from the service matching middleware. The service search engine takes a *service request* and available service *description collections* as inputs, and returns matching services as well as their matching degrees.

## 5 Grid Service Description

### 5.1 Service IOPEs

Inputs, outputs, preconditions and effects (IOPEs) are important functional attributes for a web/Grid service. Inputs and preconditions define the constrains required for a service invocation, and outputs and effects indicate the results or the state transformation of a service execution. As a standard web service description language, OWL-S (Martin et al. 2004) provides an all-side ontology classes definition for describing a service's IOPEs. A number of semantic approaches have adopted standard OWL-S ontology class structure to describe services and based the service discovery on the state transformation the service produces (Martin et al. 2005) (Sycara et al. 2003).

IOPEs are important characteristics for describing a service. However, in order to realize a semantic service matching for pervasive clients discovering Grid services, other types of service attributes also have to be considered.

### 5.2 Service Resources

Service-oriented Grid computing architecture is an extension of current Web Service technologies. In the computing architecture, applications are built on top of a set of common, standard and high-level services, which meet the definition of Open Grid Services Architecture (OGSA). One of the important requirements of OGSA is that underlying middleware should store information about the service state because Grid application users usually need this kind of information to be maintained from one invocation to another.

Web Service Resource Framework (WSRF) provide a mechanism of building the stateful services required by OGSA. It specifies a straightforward solution of recording the service state: keep the web service and its state information completely separate, and store all the state information in an entity named "resource". Each resource entity in a web service is assigned a unique key. When service clients want to invoke a service, they submit the request including both the URI of the service provider and the key of the required resource. A service may have several resource instances, which enables the state information to be kept for different purposes.

Service resources have different types and characteristics. They may be a integer value, keeping the value of mathematical operations; they may be a virtual shopping chart, recording items buyers choose; they may represent a physical device (e.g. printer), logging its working status. WSRF specifications define several style of interaction mechanisms, providing different ways of representing and accessing multiple resource instances. Because the service resource is a key parameter for the Grid service invocation, we regard it as an important functional attribute in the Grid service description.

### 5.3 Service Type

In a service-oriented pervasive Grid environment, pervasive devices form the intersection between the physical world and the digital world. Users execute their tasks by using a variety of Grid services through their handheld devices. Two main styles of application scenario are identified from the viewpoint of users: an information access scenario, and a work assistant scenario. In the information access scenario, the task is to collect required information or knowledge. The users' pervasive devices act as universal operating terminals to access various available Grid services. A typical example is that a doctor is able to check the data being collected from patients in real time with his/her PDA by invoking the medical Grid services deployed by the hospital.

In the work assistant scenario, users usually need to execute relatively complicated applications such as data-deluge programs to achieve specific tasks through their pervasive devices. However, due to resource limitations, most complex programs cannot be executed on a handheld device. Users have to offload resource-demanding programs of the task to the Grid, and the Grid provides the executing environment for users to achieve their tasks. A possible example is that a fire fighter may submit streams of temperature data about a multi-story building to the Grid. The Grid assists fire fighters to solve three-dimensional partial differential equations in order to obtain the detailed information such as the temperature of different floors of the building, or the temperature of a particular room. The work assistant scenario describes the scene in which pervasive users achieve complicated tasks with the assistant of Grid services.

An ontology is defined on the basis of the analysis of two application scenarios. The ontology represents a hierarchy of possible application scenarios and contains a taxonomy of service types which are usable for pervasive clients. Figure 2 shows a class diagram of the service type ontology. The top-level concept of the ontology is *Service*, which represents the most generic type. There are two direct subclasses of *Service*: the *InfoRetrieval* class represents the general service for the information access scenario; the *WorkAssistant* class represents the general service for the work assistant scenario.

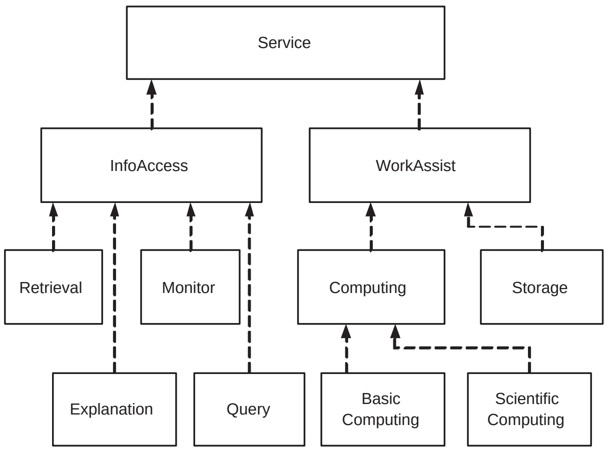

Figure 2: Service Type Ontology Diagram

### 5.4 Service Context

The vision of pervasive computing is to embed a variety of computing systems into our everyday life seamlessly, providing information and services anytime and anywhere. In a dynamic pervasive Grid environment, Grid services should be provided for pervasive users in the right way at the right time in the right place. To achieve this goal, both service consumers and service providers in the service-based Grid environment need to share their knowledge with each other. An ontology is designed to model the context of the service-oriented Grid environment (Figure

3). The top-level of the context ontology is expressed as six classes, which covers the most basic elements of the service, and together with their subclasses form a basic framework of the environment context information. The detailed design of the context ontology is discussed in our previous paper (Guan et al. 2006).

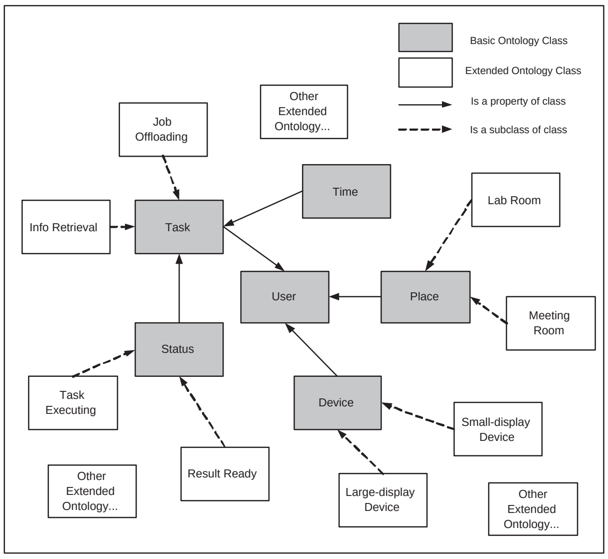

Figure 3: Context Ontology Diagram.

Service context attributes are required when describing a Grid service. At present, we consider two context attributes in the Grid service description: the first is the service location, which corresponds to the "Place" class of the environment context ontology; the other is the service access range, based on which a service discovery restricting mechanism is implemented.

pervasive users access Grid services with their portable devices, which may expose their personal information. For example, if our service directory is so "open" that every pervasive users can discover and obtain all deployed Grid services, a user location information may be exposed to other users as long as they can locate and try to invoke corresponding location-monitoring services. Hence, protecting personal privacy is an essential issue of designing a service discovery mechanism. The user personal information decides their accessing level during the authorization process, which is shown in their "User" class. The service provider defines the service range for every service in the service description. When a new pervasive user sends a request to search Grid services, the service searching engine will reason and decide whether Grid services can be exposed by comparing the service access range of Grid services and the access level of this pervasive user.

### 5.5 Service Detail

Every service has particular parameters which are important when considering the service matching. For example, for a "Printer" service, the "PaperSize" attribute specifies the supporting paper size. When a user intends to print a large-size image (e.g. A1 size), the "PaperSize" has to be considered to be a high priority matching requirement. Otherwise, the user may locate a number of useless services which do not support large-size printing. The definition of service detail attributes depends on the detailed implementation of the individual service, and because of this there is not an abstract ontology class designed for it.

### 5.6 Service Description with Extended OWL-S

OWL-S (Martin et al. 2004) is a language for describing services, which provides a standard vocabulary that can be used together with other aspects of the OWL (Smith et al. 2003) description language to create service descriptions. The structure of the OWL-S upper-level ontology is based on the types of knowledge of service description: the "Service Profile" provides the high-level descriptive information of a service, such as the name, input/output of the service, and additional text description; the "Service Model" and "Service Grounding", provide sufficient information of how the service is used and how to interact with the service.

We use the OWL-S language to describe Grid services. However, the "Service Profile" does not specify the Grid service attributes required in our pervasive Grid computing environment. Hence, it has to be extended by adding the service parameters discussed above. Figure 4 illustrates the extended service profile class and its properties.

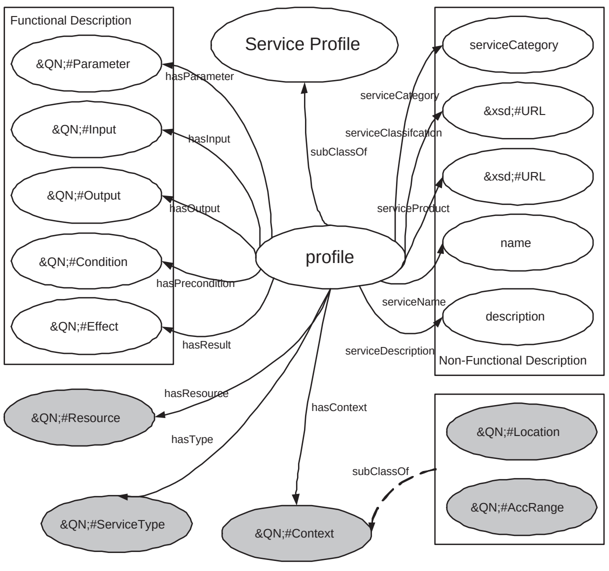

Figure 4: Extended Service Profile.

Grid services are described based on the extended service profile. For a Grid service, its description collection includes the functional attributes (e.g. inputs, outputs, preconditions, effects), the service type (an instance of *InfoRetrieval* class or *WorkAssistant* class), the service resource, the service detail, and the service context information (the location of the service provider and the service access range).

## 6 Service Discovery Algorithm

As discussed in the previous section, a service request is composed of a number of individual requirements, specifying various attributes to be expected in services. The service search engine takes a service request and a group of service description collection as inputs, and is responsible for deciding whether a Grid service is a matching service for this service request. The service search engine compares a Grid service description collection with the service request. Initially, the service search engine will check to judge whether each strict requirement can be matched precisely in the service description. If a service description does not contain the expected attributes, it will be dismissed and the service search engine will compare the next service description with the request. If a Grid service satisfies all of the strict requirements, the search engine will then turn to estimate the general requirements.

Subsumption reasoning based on the taxonomic relation between concepts is used to determine the matching degree between general requirements and related service attributes. We define three expected matching levels for every general requirement.

- "Substitute" indicates that the requirement expects to find a concept in the service description which is equal to or is the direct superclass of it.
- "Cover" indicates that a concept which subsumes the concept in the service description is expected to find to satisfy this requirement.
- "Fuzzy" means this requirement is little important for the service matching. As long as a concept in the service description which has the subsumption relationship (either superclass or subclass) with it, this requirement is satisfied.

These expected matching levels can be set when the service request is submitted to the service search engine. The service search engine will check the similarity between each general requirement in the service request and the related service attribute in the service description. The detailed similarity is determined by the semantic relationship between predefined concepts or their relationships. If all of the expected matching levels are satisfied, this service will be a reasonable candidate matching service for the service request.

The service search engine can find a number of candidate services for a specific service request. Although the service matching mechanism is not responsible for the service selection, the information about each candidate service matching degree is required to be provided as a result for the service request. We use the term "MatchingScore" to show the matching degree of the candidate service. For a candidate service, its MatchingScore is calculated using the following equation (Bandara et al. 2007):

$$MatchingScore = \sum_{i=1}^{n} Score_i/n$$

The "$Score_i$" indicates the matching degree of every individual requirement in the service request against the related service attribute in the service description. The score is obtained by calculating the semantic distance $||C_r, C_a||$ of the ontology structure between the individual requirement ($C_r$) and the related service attributes ($C_a$). We refer to the equation presented in (Caceres et al. 2006) to calculate the individual score.

$$Score_i =$$

$$\begin{cases} 1 & if\ C_a\ =\ C_r \\ \frac{1}{2} + \frac{1}{2 * e^{||C_r, C_a||}} & if\ C_a\ is\ a\ superclass\ of\ C_r \\ \frac{1}{2} * e^{||C_r, C_a||} & if\ C_r\ is\ a\ superclass\ of\ C_a \end{cases}$$

The MatchingScore of each candidate service is calculated based on $Score_i$, and it will determine the ranking of candidate services. The higher the score is, the higher ranking the candidate service has.

## 7 Implementation and Experimental Results

The ontology classes for the Grid service description are defined with the OWL language using the Protege toolkit. Protege is an open-source ontology editor and knowledge-based framework. It can also be used to create OWL-S services by integrating an OWL-S editor plug-in.

The Grid service matching middleware has been implemented with the jUDDI toolkit, the Racer system (Haarslev & Moller 2003), the MySQL database and other related techniques. Figure 5 shows its internal components. The middleware is written as both a Java Web Service for use by other middleware in the system architecture and a web application using the AJAX design mode which can accessed through a standard web interface.

When a Grid service is published through service publishing port, the service will be forwarded to the publishing manager. The service description contains the semantic information. The publishing manager classifies the semantic information, including the service context, the service functionality, the service type by using the Racer reasoning system, and store the service advertisement in the repository.

The service matching middleware also provides a service query port, which can be used to search a service based on its capabilities. The service query is transmitted to query manager. Once the semantic matching based on the extended OWL-S profile information is completed, a list of matching services will be returned to service discovery manager. In additional to the candidate services, the matching score for each candidate service will also be attached to the response. The request will determine to select the appropriate Grid services for the task execution.

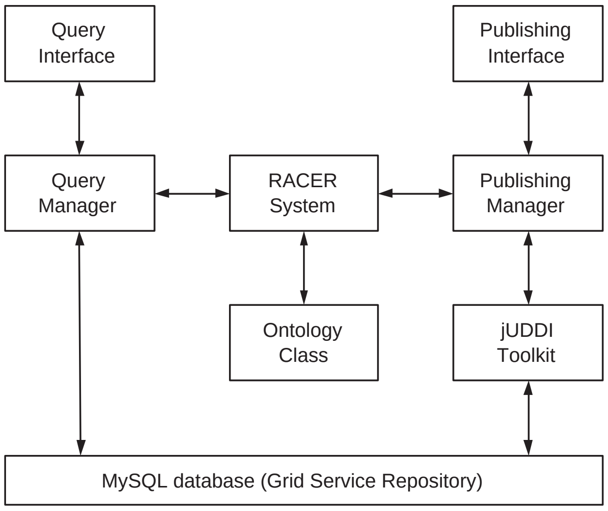

Figure 5: Internal Components of Service Matching Middleware.

To evaluate our service matching middleware, we measured the performance of our service matching middleware against a number of service requests.

The following experiment was designed in order to obtain the time of querying a Grid service. Both the description information of real Grid services and a large number of pseudo services are published in the service repository. Altogether, more than fifty services can be accessed by the semantic service matching middleware. A UDDI web service registry was built and a number of web services is published onto it. Table 1 shows the average time of querying a service on two different service discovery platform. The time of querying a Grid service with semantic concepts is longer, because the additional computation effort is required to determine the concept subsumption relationships in the logic reasoning system.

| | UDDI | Semantic Matching Middleware |
|---|---|---|
| Time (ms) | 37.4 | 52.1 |

Table 1: Time of Querying a Service

Although UDDI has a faster system querying performance than our semantic service matching middleware, it has several shortcomings when used in practice for the service discovery. UDDI does not provide sufficient technical details of the service, does not support any inference based on the concepts, can only support the search based on the string comparison, and cannot identify a match between functionally equivalent services that are described by different key words. Our service matching middleware overcomes these shortcomings by using the semantic service description and discovery mechanism. We believe it is worth obtaining a relatively-significant improvement in system function at the price of a small increase in the service discovery time.

In the above experiment, the time of returning one service only is measured. To test the system scalability, we used five kinds of service repository sizes (10, 20, 50, 100, and 200) and varied the number of matching services to be one, two, four,or eight. Figure 6 shows the experimental results. As we expected, the system response time is within an acceptable limit and loosely proportional to the size of the service repository and the number of the matching results.

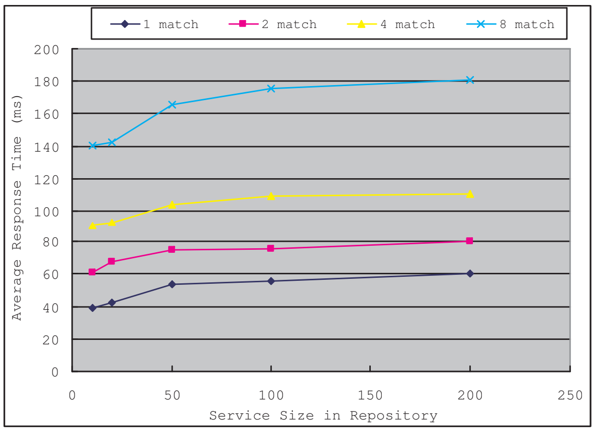

Figure 6: Average Query Time of Service Matching Middleware.

## 8 Conclusion

In a service-oriented Grid environment, pervasive clients are concerned about three aspects when considering the problem of using Grid services to perform their tasks. The first is a method that describes what capabilities Grid services support is required to be developed so that services can be advertised to provide contributions for the task achievement. The second is building a Grid service discovery mechanism so that services can be located by pervasive users. The third is the mechanism of Grid service invocation so that required information or resources can be provided during the process of the task execution. Essentially, Grid service description, discovery and execution are interdependent: Grid service description is the prerequisite of Grid service discovery; the mechanism of Grid service discovery determines how a Grid service should be described; the service execution process depends on the discovery of Grid services.

In this paper, we have presented a semantic service matching middleware for the service-oriented pervasive Grid environment. A number of service attributes have been defined to represent service characteristics in the service description. Because of the centralized range of Grid service registry in the system infrastructure, a service search engine is built with the extended OWL-S semantic language and the RACER ontology reasoning system, providing query interfaces for users or other middlewares to locate required services. The service request is compared with a service description collection in two steps:

strict requirements have to be matched precisely; general requirements are checked based on the user-expected matching level. The matching services as well as their matching degree are returned as the result for a service request. The semantic service matching middleware has been integrated into our system architecture and it interacts properly with other middlewares. We also measure its performance against a number of service request, and the results show that only a small increase in the service discovery time compared to the traditional service discovery mechanism while a significant improvement of the system function has been obtained.

In the future, we plan to continue current research work to allow such a Grid service matching mechanism to be extended so that it can be more suitable for the service-oriented pervasive Grid environment. OWL-S is a language for describing services, providing a standard vocabulary that can be combined with the other aspects of the OWL description language to create service descriptions. It supports not only automatic service discovery, but also automatic service invocation, composition and interoperation. At present, we only use the "Profile model" to describe Grid services. In the future, we will extend to use both the "Process model" and the "Grounding model" to build Grid service descriptions, enabling the vision of the automatic service discovery, composition, and invocation to be realized.

## REFERENCES

Bandara, A., Payne, T., Roure, D. D. & Lewis, T. (2007), A semantic approach for service matching in pervasive environments, *in* '2nd International Workshop on Pervasive Systems', p. Portugal.

Bhagyavati & Kurkovsky, S. (2003), Wireless grid enables ubiquitous computing, *in* 'the International Conference on Parallel and Distributed Computing Systems'.

BluetoothSIG (2001), 'Specification of the bluetooth system – core'.
**URL:** *http://www.bluetooth.org/*

Caceres, C., Fernandez, A., Ossowski, S. & Vasirani, M. (2006), 'Agent-based semantic service discovery for healthcare: An organizational approach', *IEEE Intelligent Systems* pp. 11–19.

Guan, T., Zaluska, E. & Roure, D. D. (2006), Extending pervasive devices with the semantic grid: A service infrastructure approach, *in* 'Sixth IEEE Conference on Computer and Information Technology'.

Haarslev, V. & Moller, R. (2003), Racer: a core inference engine for the semantic web, *in* 'Proceedings of 2nd International Workshop on Evaluation of Ontology-based Tools', pp. 27–36.

Kurkovsky, S., Bhagyvatim & Yang, M. (2004), Medeling a grid-based problem solving environment for mobile devices, *in* 'the International Conference on Information Technology: Coding and Computing', pp. 05–07.

Li, L. & Horrock, I. (2003), A software framework for matchmaking based on semantic web technology, *in* 'In. Proc. 12th Int World Wide Web Conference Workshop on E-Service and the Semantic Web'.

Majithia, S., Shaikh, A., Rana, O. & Walker, D. (2004), Reputation-based semantic service discovery, *in* 'Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises'.

Martin, D., Burstein, M. & etc., J. H. (2004), 'Owl-s: Semantic markup for web services'.
**URL:** *http://www.w3.org/Submission/OWL-S*

Martin, D., Paolucci, M., Mcilraith, S. & Burstein, M. (2005), Bringing semantics to web services: The owl-s approach, *in* 'SWSWPC'.

MicrosoftCorporation (2000), 'Universal plug and play device architrcture'.
**URL:** *http://www.upnp.org/*

Millard, D. E., Woukeu, A., Tao, F. & Davis, H. C. (2005), Experience with writing grid clents for mobile devices, *in* '1st International ELeGI Conference'.

OASIS (2005), 'Uddi sepcifications'.
**URL:** *http://www.uddi.org/specification.html*

Roure, D. D., Hey, T. & Trefethen., A. E. (2006), 'A global e-infrastructure for e-science - a step on the road to ambient intelligence', *Chapter in "True Visions: Tales on the Realization of Ambient Inteligence", Edited by E.H.L. Arts and J.L.Encarnacao, Springer* pp. 209–229.

Smith, M., Welty, C. & McGuinness, D. (2003), 'Web ontology language guide version 1'.
**URL:** *http://www.w3.org/TR/owl-guide*

Srinivasan, N., Paolucci, M., & Sycara, K. (2006), Semantic web service discovery in the owl-s ide, *in* 'Proceedings of the 39th Hawaii International Conference on System Sciences'.

SunMicrosystems (2001), 'Jini: Architecture specification'.
**URL:** *http://www.sun.com/software/jini/specs/*

Sycara, K., Paolucci, M., Ankolekar, A. & Srinivasan, N. (2003), 'Automated discovery, interaction and composition of semantic web services', *Journal of Web Semantics* pp. 27–46.