Semantic Discovery for Context-Aware Service Provisioning in Mobile Environments

Alessandra Toninelli, Antonio Corradi, Rebecca Montanari

Dipartimento di Elettronica, Informatica e Sistemistica
Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna - Italy
{atoninelli, acorradi, rmontanari}@deis.unibo.it

Abstract. Advances in telecommunication and wireless systems together with the increasing diffusion of portable devices are enabling a pervasive and ubiquitous computing infrastructure for service provisioning, where mobile users expect to access the needed data from ubiquitous attachment points and require context-aware services, i.e., services that can adapt provided results to changing context information, such as variations in user relative position, in user requirements, and in locally available resources. A crucial requirement for the provisioning of context-aware services is the dynamical retrieval and interaction with local resources, i.e., discovery. Traditional discovery solutions, which are essentially based on the exact matching of syntactic patterns, like for instance identifiers, interfaces or keywords, cannot be flexible enough to effectively deal with the heterogeneity typical of mobile and pervasive environments. Recent research efforts that exploit semantic, i.e., meaning-based, techniques for the discovery of services have emerged and proved to be able to overcome this kind of limitations. However, semantic-enabled discovery solutions still seem to be underestimated within the pervasive community. In this paper we propose a middleware that supports the intelligent discovery of context-aware services for mobile users, by providing an automatic tool that enable users to specify service queries at a high level of abstraction and a semantics-based matching functionality.

1 Introduction

Recent advances in the computational capabilities of commonly used portable devices, like cellular phones and palmtops, together with their increasing ability to wirelessly communicate with other devices has favoured the development of pervasive and ubiquitous computing infrastructures for service provisioning. In this scenario mobile users are willing to access the needed services from ubiquitous attachment points and even when changing physical locations, e.g. at their workplaces, at homes or at publicly accessible places like airports and shopping malls. Moreover, users expect to be provided with context-aware services, i.e., services that can adapt provided results to changing context information, such as variations in user relative position, in user requirements, and in locally available resources [1].

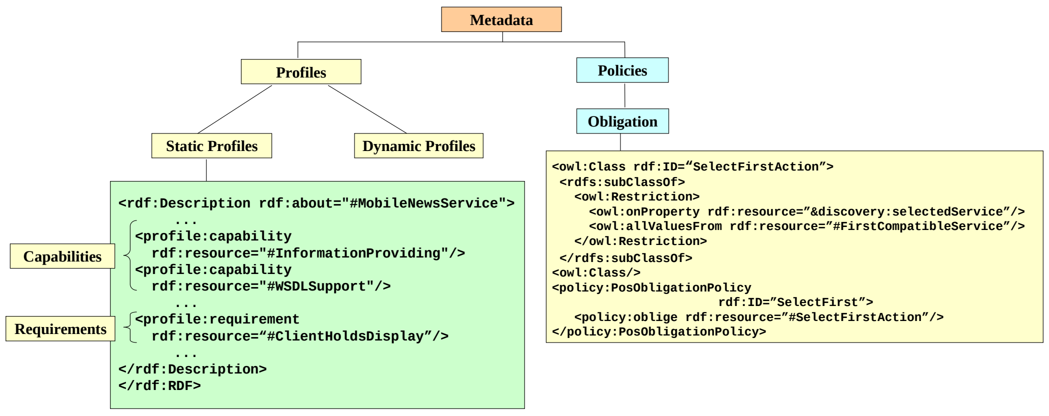
In such a scenario, the activity of dynamically retrieving and interacting with the resources available in the network vicinity without assuming a deep knowledge at the client side, i.e. the *discovery* activity, represents a crucial capability for enabling the provisioning of context-aware services. In the past few years, industry and academia have investigated several discovery solutions and this research efforts result in the proliferation of various discovery protocols like for instance the Bluetooth Discovery Protocol [2], the UPnP Simple Service Discovery Protocol [3], the IETF Service Location Protocol [4] and the Jini [5] discovery architecture, which is specifically bounded to the Java programming language. All these protocols use similar description and matching techniques, which exploit patterns, i.e. unique identifiers, interfaces or names. Other discovery protocols that have emerged within the Web Services research community, namely ebXML [6] and UDDI [7], rely on the exact matching of XML-based keywords, generally defined within fixed, standard taxonomies.

However, the use of exact matching of patterns or keywords does not represent a suitable discovery solution for mobile and pervasive architectures because it essentially lacks of the flexibility needed to deal with such an heterogeneous environment. In fact, many discovery attempts are likely to fail simply because of a syntactical mismatching between service names. For instance, if a user is looking for a “News” service, a service called “Information” would not match with the request, although its *meaning* is perfectly compatible with the meaning of the requested service. In order to overcome the limitations of traditional discovery models, the adoption of Semantic Web languages to describe and retrieve resources has recently gained considerable attention. The main advantage deriving from the adoption of semantic languages lies in the fact that they permit explicit context representation at a high level of abstraction while enabling automated reasoning about this representation. Some research efforts have already emerged in the field of Web Services that intend to enhance current discovery protocols with semantic-based description and matching techniques [8], [9].

We believe that, until now, the potential of Semantic Web techniques applied to discovery in pervasive environments has not been fully exploited. In fact, among the several research efforts that are concerned on discovery solutions for ubiquitous and ad-hoc networks [10], [11], [12], only few research proposals have addressed the problem of semantically enhancing resource discovery for mobile and ubiquitous applications [13][14]. In this paper we propose MIDAS (**M**iddleware for **I**ntelligent **D**iscovery of context-**A**ware **S**ervices), a middleware that supports semantic discovery for the provisioning of context-aware services to mobile users. This middleware enables mobile users to express their service requirements at a high level of abstraction, and performs service discovery basing on the semantic matching between user request and service offer. MIDAS integrates with the Java-based CARMEN system, which supports the provisioning of context-dependent services to portable devices [1]. The paper is organized as follows. Section 2 describes the configuration model adopted in our framework. Section 3 presents an overview of the middleware architecture. The usability and effectiveness of the proposed middleware is evaluated through the discussion of a case study in Section 4. Concluding remarks and future research directions are given in Section 5.

2 MIDAS Metadata Model

In order to perform the discovery activity, it is essential to describe resources and entities so that they can be advertised and retrieved. MIDAS adopts metadata to represent interacting entities at a high level of abstraction and to specify the desired configuration settings to apply in guiding the discovery process. In particular, MIDAS distinguishes two kinds of metadata: profiles and policies (see Figure 1).



```
<rdf:Description rdf:about="#MobileNewsService">
       ...
  <profile:capability
    rdf:resource="#InformationProviding"/>
  <profile:capability
    rdf:resource="#WSDLSupport"/>
       ...
  <profile:requirement
    rdf:resource="#ClientHoldsDisplay"/>
       ...
</rdf:Description>
</rdf:RDF>
```

```
<owl:Class rdf:ID="SelectFirstAction">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty rdf:resource="&discovery:selectedService"/>
     <owl:allValuesFrom rdf:resource="#FirstCompatibleService"/>
   </owl:Restriction>
 </rdfs:subClassOf>
<owl:Class/>
<policy:PosObligationPolicy
                      rdf:ID="SelectFirst">
   <policy:oblige rdf:resource="#SelectFirstAction"/>
</policy:PosObligationPolicy>
```

Figure 1. MIDAS Metadata Taxonomy

Profiles are used to describe the characteristics of services, users and devices. In this paper we focus on service profiles. For further details on user and device profile please refer to [15]. A service is described by a static profile and a dynamic profile. The *static* profile contains data that typically remain unchanged during service provisioning. Only in case of significant changes in the service semantics or in some of its technical characteristics, e.g., an upgrade that enables new features, the static profile must be modified. The static profile contains information about the functionalities the service provides, e.g., printing, news or language translation, and about how these functionalities are achieved, e.g., supported interfaces and communication protocols. These kinds of information represent the *capabilities* of a service. In addition, the static profile includes the service *requirements*, i.e. the conditions imposed by the service in order to be accessed. For example, a video-on-demand service may require to be properly used that the client device holds a display.

On the other side, the *dynamic* profile describes service properties that are likely to change, either due to a spatial movement of the service, which may migrate from one locality to another, or simply as a consequence of the service ongoing activity. In particular, the dynamic profile includes the grounding information part and the state part. The *grounding* of a service represents its concrete binding and implementation, e.g., the endpoint where the service can be actually invoked at, and the invocation signature. The *state* of a service includes several elements that characterize its dynamic operating conditions, like for instance the number of currently opened sessions, the average response time or more basic information like availability and liveness.

MIDAS adopts policies to express the configuration choices that rule the discovery process. In particular, MIDAS uses obligation policies to state which configuration actions have to be carried out at certain event occurrence, given that specific conditions are verified. For instance, if a mobile user has stated that she is looking for any locally available "news" service, then, at the first discovery of a service that is compatible with the user request, the semantic discovery process will terminate without any further research.

MIDAS exploits Web Ontology Language (OWL) [16] to express both static profiles and policies. The main advantage offered by OWL is that it enables to formally determine the meaning of each metadata term, so that it can be automatically acquired and processed. Semantics is assigned to metadata by means of dedicated ontologies, which represent the specification of the model describing a resource, for example a device or a service. In particular, to model and express policies, MIDAS adopts the semantics of KAoS policy ontology [17].

3 MIDAS Middleware

MIDAS provides the functionalities needed to perform a service discovery process based on the abstract requirements expressed by the user. In particular, it enables users to specify a service query at a high level of abstraction, and it subsequently reasons about the query in order to verify its compatibility with locally available services.

MIDAS is built to integrate with the Java-based CARMEN system that supports the provisioning of context-dependent services to portable devices [1]. CARMEN provides any portable device with a companion middleware proxy (*shadow* proxy) that autonomously acts on its behalf over the fixed network and follows user/device movements among network localities. CARMEN implements shadow proxies by exploiting the Mobile Agent programming paradigm [18]. In particular, CARMEN provides proxies with execution environments, called places, that typically model nodes. Places can be grouped into domains that correspond to network localities, e.g., either Ethernet-based LANs or IEEE 802.11b-based wireless LANs. MIDAS interacts with the two main components of CARMEN middleware, namely the Metadata Manager (MM) and the Context Manager (CM). MM supports the specification, modification, checking for correctness, installation and evaluation of different kinds of profiles and policies. CM dynamically determines the context of a CARMEN client, i.e. the logical set of resources accessible to the client during a service session, and manages services bindings in case of context modifications. The client context is determined on the basis of user's preferences, device characteristics and current conditions that hold within the locality. CM is also in charge of providing users with the references to needed services and resources, as users are not allowed to directly access resources. More detailed information about CARMEN middleware are provided in [1].

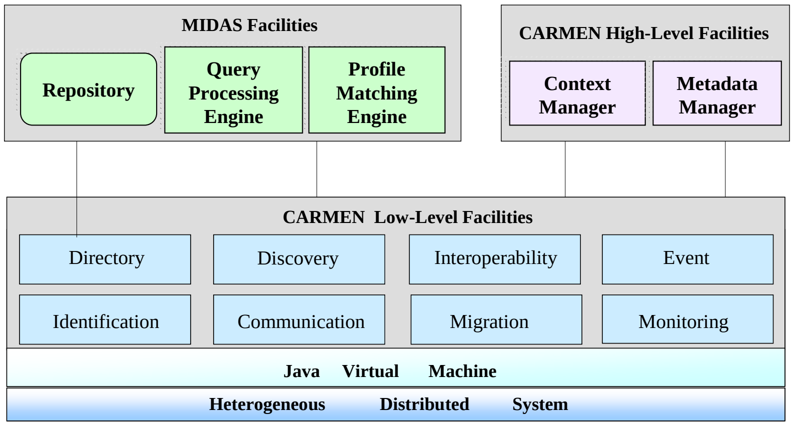


Fig. 1. MIDAS Middleware Facilities

MIDAS middleware components are shown in Figure 2. The **Repository** stores profile information about services, users and devices. In particular, in this repository MIDAS loads at service start-up the static profile and the grounding information about each locally available service. On the contrary, the state part of the dynamic profile is not stored in the repository, but is autonomously maintained by each service. The criteria behind this allocation choice are mainly related with efficiency issues. In fact, as the state of a service is likely to change rather frequently, the prompt updating of a repository would require a considerable amount of communication resources that may degrade system performance. In addition, the repository is exploited to store OWL service ontologies, which are needed for reasoning during the discovery process. New service profiles and ontologies may also be loaded during service provisioning, given that no discovery process is currently being executed. The repository is implemented via the CARMEN directory facility.

The **Query Processing Engine** (QPE) is in charge of processing the user request in order to obtain a list of requirements, which describes the service characteristics the user is looking for. Once the user has specified her desired service characteristics, this information is translated by a dedicated parser into a static profile template. The template consists of a list of user requirements, i.e. of capabilities that the user considers mandatory for selecting a service, and a list of user preferences, i.e. of capabilities that the user would like the selected service to exhibit, although they are not considered mandatory. For example, a user may be looking for a news service that permits the reading and browsing of newspaper data resources available over the fixed Internet. In this case, the user may state that she wants to read “The Times” online (requirement), while she may state that she would like to have a look at the “Corriere della Sera”, if available (preference). In addition, QPE receives from the shadow proxy the user and the user’s device profile. The information provided by the user/device profile is exploited to customize QPE behaviour on the basis of the information provided with the profiles. For instance, let us suppose that a user has stated in her profile that she can speak two languages, e.g. English and Italian. When this user asks for a translation service, QPE automatically narrows the request field by offering to the user choice only translation services that concerns either English or Italian.

Finally, user preferences concerning configuration settings, like for example the automatic invocation of the “best” service, are transformed into KAoS policies and subsequently enforced during the semantic discovery phase in order to rule the middleware components’ behaviour. Our current QPE implementation provides a graphically interfaced automatic tool that assists the user in formulating simple service requests and in specifying discovery configuration settings (see Figure 3).

The static profile template is then passed by QPE to the **Profile Matching Engine** (PME), which is responsible for performing a step-by-step matching algorithm between user requirements/preferences and service capabilities. After having received a profile template from QPE, PME first asks to CARMEN Context Manager the user context, i.e. the logical set of user-accessible resources. In particular, PME selects the accessible services and, for each of them, it retrieves the static profile via the CARMEN directory facility. The acquired information, together with the reasoning capabilities provided by Jena and the service ontologies stored within the domain, are exploited to perform a matching algorithm.

The matching algorithm works on one service at a time. For each user requirement, PME verifies if the service profile contains one or more compatible capabilities. The concept of compatibility covers several possibilities, ranging from the simplest case of an exact matching between the required and the offered capability, to less direct relationships, e.g., the case of a service capability which is logically included by the user requirement. For example, if a user requires a “printing” capability, a “colour_printing” capability may be well suited to fulfill the user request. Note that indirect matching cases are recognized by means of semantic reasoning that, in the current implementation, is provided by Jena semantic framework [19]. In order to distinguish the different types of match, we have adopted a scoring system, where direct matches are assigned the highest score. At present PME is able to assign only two possible scores, i.e. direct or indirect match, but we are planning to extend this approach by means of a more articulated scoring function. In case of complex requirements, e.g. the OR of two requirements, some additional logic is needed in order to correctly compare service capabilities and user requirements. For example, in the OR case, the simple matching algorithm is first executed on one requirement and, only in case of failure, also executed on the second requirement.

Depending on current configuration settings, PME may either stop executing the matching algorithm at the first occurrence of a compatible service (“select first” modality), or perform the algorithm on each service included within the user context (“complete” modality). In the first case, PME returns to CM a single service, while in the latter case it returns a list of services, ordered on the basis of the matching score assigned. Let us note that MIDAS is not bounded to any specific discovery protocol since its semantic functionalities are conceived to lay at a higher level of abstraction.

In the present implementation MIDAS relies on the Jini-based CARMEN discovery facility.

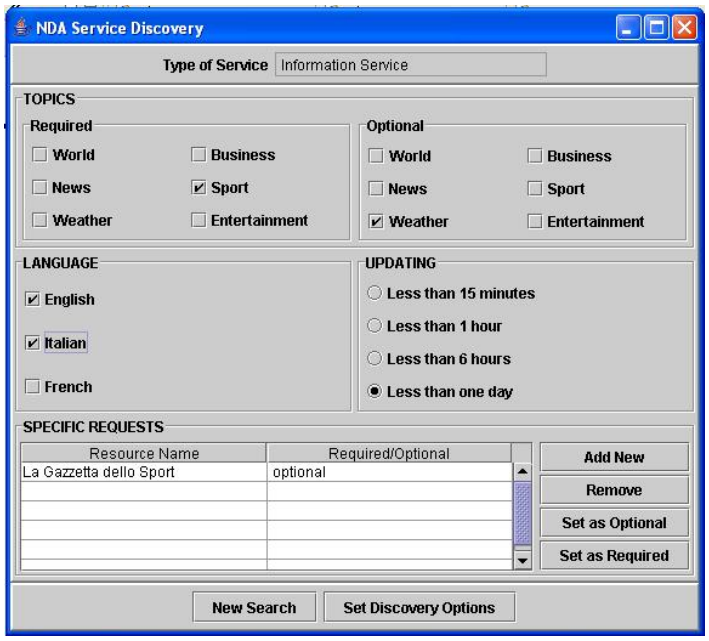
4 Case Study

We have tested MIDAS in the design and implementation of a prototype News Discovery Assistant (NDA) that enables mobile users to access information services available on the wired Internet, like for instance newspaper reading services, news agencies access services or radio/television digital broadcasting services.

Our simulated testbed setting for NDA consists of a wireless metropolitan network composed by several 802.11 network localities, with each locality modeled as a CARMEN domain. Each domain provides execution environments (*places*) for shadow proxies on each physical node, offers CARMEN and MIDAS middleware facilities and hosts service components providing information about the locally available resources. Resources, entities and services are described by means of semantic metadata. Each place hosts an instance of both CARMEN Metadata and Context Manager, whereas the Query Processing Engine and the Profile Matching Engine are implemented as centralized elements that reside only on one predefined place within the domain. In addition to this, each domain is provided with a repository.

The NDA service is accessible from wireless devices through several access points, which may be located, for example, at the airport, at the railway station, in the hall of a shopping centre or in other publicly accessible places. NDA users interact with the MIDAS infrastructure via device-specific clients running on their wireless access devices. Client applications enable users to subscribe to NDA by filling in a form with user profile and to authenticate themselves to the service before starting any NDA session. When a user first accesses the service, NDA instantiates a shadow proxy in the domain where the user is currently attached. At service provision time, the clients are only in charge of forwarding user requests (and of visualizing the received service results) to (from) their responsible proxies. Whenever a new user wishes to start a semantic-driven discovery of locally available services, the MIDAS middleware initiates a discovery process by interacting with both the user proxy and CARMEN Context Manager. In particular, the user proxy forwards a request for service to MIDAS, which parses both the query and the candidate services' profiles. Then, it exploits the acquired information, along with its reasoning capabilities, to find if one or more services fulfill the requested characteristics. The selected services are finally passed to CM, so that the user can access them.

Let us suppose that Alice, while waiting for her train at the railway station, is willing to access a news service using her 802.11b enabled palmtop. In particular, Alice would like to download today sport news from available newspapers so that she can read them during her journey. Once seated in the waiting room, Alice exploits the wireless connectivity provided within the station area in order to access NDA. As Alice first connects from the station network area, NDA instantiates a shadow proxy within the station CARMEN domain. The device-specific client allows Alice to specify her profile, her device profile and her setting preferences, which will be retrieved at the beginning of any new NDA session from any CARMEN domain. On the basis of profile metadata and of Alice location, the Context Manager dynamically determines Alice context. For instance, as Alice states in her profile that she can speak English, Italian and French, CM discards from the list of locally available information resources all the services that are provided in neither of these languages. Also, the device profile is used to select only resources that are technically compatible with Alice palmtop. For example, services not supporting 802.11b are not included in the context. In addition, setting preferences are translated into policies to be used during service provisioning in order to customize NDA behaviour. In particular, Alice asks for a "select first" discovery process, which terminates at the first discovery of a request-compatible service, and selects the automatic modality for service invocation.


NDA Service Discovery

Type of Service: Information Service

TOPICS

Required: World, Business, News, [✓ Sport], Weather, Entertainment

Optional: World, Business, News, Sport, [✓ Weather], Entertainment

LANGUAGE: [✓ English], [✓ Italian], French

UPDATING: Less than 15 minutes, Less than 1 hour, Less than 6 hours, [✓ Less than one day]

SPECIFIC REQUESTS

Resource Name	Required/Optional
La Gazzetta dello Sport	optional

Figure 2. NDA Prototype GUI

After the initialization phase, Alice is allowed to forward to NDA a service request using the graphic interface shown in Figure 3. The GUI contains different panels that specifically refer to a NDA service, i.e., an information service. By selecting the appropriate fields within the various GUI panels, Alice is enabled to describe in a very intuitive manner the kind of service she is looking for. For instance, she specifies that she is mostly interested in reading sport news (mandatory requirement), but would also like to have a look at the weather forecasts (optional requirement). The second panel, which is customized on the basis of the user profile, let Alice express her choice about the language of the desired service. For example, Alice selects from the list of possible languages only English and Italian because she is not interested in reading French newspapers. In addition, as she is looking for today news, Alice requires to be provided with news that are at least daily updated. In the last panel, Alice enters the name of a particular information resource she would like to access, namely the sport newspaper “La Gazzetta dello Sport”.

```
<rdf:RDF>
              ...
<profile:Capability rdf:ID="ProvidesSportInfo">
  <owl:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&nda_service:infoType"/>
      <owl:allValuesFrom rdf:resource="&nda_service:Sport"/>
    </owl:Restriction>
  </owl:subClassOf>
</profile:Capability>
<profile:Capability rdf:ID="DailyUpdated">
  <owl:subClassOf>
     <owl:Restriction>
        <owl:onProperty rdf:resource="&nda_service:updateFrequency"/>
        <owl:allValuesFrom rdf:resource="&nda_service:OneDay"/>
     </owl:Restriction>
  </owl:subClassOf>
</profile:Capability>
<profile:Capability rdf:ID="EnglishOrItalianLanguage">
  <owl:UnionOf>
    <owl:subClassOf>
       <owl:Restriction>
          <owl:onProperty rdf:resource="&nda_service:language"/>
          <owl:someValuesFrom rdf:resource="&nda_service:English"/>
      </owl:Restriction>
    </owl:subClassOf>
    <owl:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="&nda_service:language"/>
         <owl:someValuesFrom rdf:resource="&nda_service:Italian"/>
      </owl:Restriction>
    </owl:subClassOf>
  </owl:unionOf>
</profile:Capability>
              ...
<profile:Capability rdf:ID="Available_LaGazzettaDelloSport">
  <owl:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="&nda_service:available_resources"/>
       <owl:someValuesFrom rdf:resource="&nda_resources:LaGazzettaDelloSport"/>
    </owl:Restriction>
  </owl:subClassOf>
</profile:Capability>

<rdf:Description rdf:about="#Template812">
   <profile:simpleRequirement rdf:resource="#ProvidesSportInfo"/>
   <profile:simpleRequirement rdf:resource="#DailyUpdated"/>
      ...
   <profile:complexRequirement rdf:resource="#EnglishOrItalian"/>
      ...
   <profile:preference rdf:resource="#Available_LaGazzettaDelloSport"/>
</rdf:Description>

</rdf:RDF>
```

(a)

```
<rdf:RDF>
<nda:ServiceProfile rdf:ID="MNSProfile">
</nda:ServiceProfile>
<profile:Capability rdf:ID="ProvidesNewspapers">
   <owl:subClassOf>
     <owl:Restriction>
        <owl:onProperty rdf:resource="&nda_service:resourceType"/>
        <owl:allValuesFrom rdf:resource="&nda_service:Newspapers"/>
     </owl:Restriction>
   </owl:subClassOf>
</profile:Capability>
<profile:Capability rdf:ID="EnglishLanguage">
   <owl:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource="&nda_service:language"/>
       <owl:allValuesFrom rdf:resource="&nda_service:English"/>
     </owl:Restriction>
   </owl:subClassOf>
</profile:Capability>
<profile:Capability rdf:ID="Available_EnglishNewspapers">
   <owl:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource="&nda_service:available_resources"/>
       <owl:someValuesFrom rdf:resource="&nda_resources:EnglishNewspapers"/>
     </owl:Restriction>
   </owl:subClassOf>
</profile:Capability>
            <!--Service profile-->
<rdf:Description rdf:about="#MNSService">
   <profile:capability rdf:resource="#ProvidesNewspapers"/>
   <profile:capability rdf:resource="#EnglishLanguage"/>
   <profile:capability rdf:resource="#Available_EnglishNewspapers"/>
            ...
</rdf:Description>
</rdf:RDF>
```

(b)

Figure 3. Service Template and MNS Profile

The information entered by Alice through the GUI is now translated by QPE parser into a service template, which is shown in Figure 4a. The first part of the service template contains a series of requirements, e.g. the service must provide sport information and must provide daily updated news (simple requirements). The complex requirement that admits both services provided in English and services provided in Italian has been automatically produced by QPE parser. The second part of the template lists Alice preferences, e.g. she would like to have a look at the weather forecasts and she would like to read “La Gazzetta dello Sport”, if available.

This service template is passed from QPE to PME, which initiates to execute its matching algorithm on the list of services currently included in Alice context. PME finds out that a local Mobile News Service (MNS), which permits the reading and browsing of some locally available newspapers, is compatible with Alice request. Figure 4b shows an excerpt of MNS static profile. In particular, MNS provides news about several topics, including sport and weather forecasts, and is provided in English. However, “La Gazzetta dello Sport” is not included in the group of newspapers accessible via MNS because MNS only offers locally available newspapers, i.e., English newspapers. Nevertheless, the service results to be compatible with Alice request as this was not a requirement, but only a preference. According to the "select first" modality, PME passes to CARMEN CM the identifier for MNS and suspends its task. CM automatically retrieves MNS grounding information via the CARMEN directory facility, and accesses the service state on the endpoint provided with the grounding information, in order to check if the service is currently available. For example, if MNS service had already reached its maximum number of opened sessions, then CM would not allowed to invoke it. In this case, CM would have to restart the semantic matching process to find another compatible service. If no available service was found, the process would terminate with a fail result.

Let us suppose that Alice is lucky and MNS service is currently available. In this case, CM tells PME to definitely end its task and passes to the user proxy a reference to directly invoke the service. After the service has been invoked by the user proxy, Alice can visualize and download on her palmtop the news she was looking for.

5 Conclusions and Ongoing Work

Resource discovery represents a crucial activity within pervasive environments where mobile users expect to be provided with context-aware and location dependent services. Semantic languages seem to represent a suitable means to realize advanced discovery solutions that can overcome the intrinsic limitations of traditional models. We propose a middleware that exploits semantic techniques to perform the discovery of context-aware services on the basis of the abstract requirements expressed by mobile users.

The MIDAS middleware still needs further improvements. We are planning to further develop the matching algorithm, by realizing a more advanced compatibility scoring function that permits the classification of services on the basis of customizable criteria. Moreover, we intend to add to the current algorithm a "reverse matching" phase that verifies the compatibility between service requirements and user/device capabilities. In addition, we are working on integrating security features in the service discovery process in order to perform the semantic matching also on security properties of users and services.

References

1. P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli, "Context-Aware Middleware for Resource Management in the Wireless Internet", *IEEE Transactions on Software Engineering*, Vol. 29, No. 12, December 2003.
2. Bluetooth White Paper. World Wide Web, http://www.bluetooth.com/developer/whitepaper.
3. UPnP White Paper. World Wide Web, http://upnp.org/resources.htm.
4. IETF Service Location Protocol. http://www.ietf.org/html.charters/svrloc-charter.html
5. K. Arnold, A. Wollrath, B. O'Sullivan, R. Scheifler, and J. Waldo, *The Jini Specification*. Addison-Wesley, Reading, MA, USA, 1999.

6. ebXML Registry Information Model v2.1, June 2002, http://www.ebxml.org/specs/ebRIM.pdf
7. The UDDI Technical White Paper. http://www.uddi.org/, 2000.
8. A. Dogac, Y. Kabak, and G. B. Laleci, Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery. Proc. 14th Int. Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04), IEEE Computer Society Press, pp.69-76, March 2004.
9. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, Importing the Semantic Web in UDDI, *Web Services, E-Business and Semantic Web Workshop*, Springer Verlag, Toronto, Ontario, Canada, 2002.
10. O.V. Ratsimor, D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, Service Discovery in Agent-based in Pervasive Computing Environments, *Journal on Mobile Networking and Applications*, November 2003.
11. C. Lee, A. Helal, N. Desai, V. Verna, and B. Arslan, Konark: A System and Protocol for Device Independent, Peer-to-Peer Discovery and Delivery of Mobile Services, *IEEE Transactions on Systems, MAN, and Cybernetics-Part A: Systems and Humans*, Vol.33, No.6, November 2003.
12. F. Zhu, M. Mutka, and L. Ni, Splendor: A Secure, Private, and Location-Aware Service Discovery Protocol Supporting Mobile Services, *Proc.1st Int. Conf. on Pervasive Computing and Communications (PerCom'03)*, IEEE Computer Society Press, Texas, USA, 2003.
13. S. Avancha, A. Joshi, and T. Finin, Enhancing the Bluetooth Service Discovery Protocol. Technical Report, University of Baltimore County, August 2001, TR-CS-01-08.
14. D. Chakraborty *et al.*, DReggie: Semantic Service Discovery for M-Commerce Applications, *Proc. Workshop on Reliable and Secure Applications in Mobile Environment, In Conjunction with 20th Symposium on Reliable Distributed Systems (SRDS)*, October 2001.
15. A. Corradi, R. Montanari, D. Tibaldi, and A. Toninelli, A Context-centric Security Middleware for Service provisioning in Pervasive Computing, *Proc. 2005 International Symposium on Applications and the Internet (SAINT2005)*, IEEE Computer Society Press, Trento, Italy, February 2005.
16. F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein, OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-ref/.
17. A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, J. Lott, KAoS policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement, *Proc. of the IEEE 4th Int. Workshop on Policies for Distributed Systems and Networks*, POLICY 2003, 4-6 June 2003.
18. A. Fuggetta, G.P. Picco, G. Vigna, Understanding Code Mobility, *IEEE Transactions on Software Engineering*, Vol. 24, Issue 8, August 1998.
19. The Jena Semantic Web Framework, http://jena.sourceforge.net/.