Dynamic Configuration of Semantic-based Service Provisioning to Portable Devices

Antonio Corradi **Rebecca Montanari** **Alessandra Toninelli**

Dipartimento di Elettronica, Informatica e Sistemistica (DEIS)
University of Bologna
Viale Risorgimento 2, 40136 Bologna, Italy
{acorradi, rmontanari, atoninelli}@deis.unibo.it

Abstract
Context-awareness is starting to emerge as a key driving principle for the design and provisioning of pervasive services in pervasive computing environments. Semantic languages seem to provide effective means to express, process and reason about context information and to facilitate knowledge sharing and interoperability among previously unknown entities. However, the exploitation of semantic languages for the design and deployment of context-aware applications requires to address several issues. In particular, semantic languages require complex and heavyweight support facilities, e.g., reasoning engines, ontology repositories and knowledge management tools, that may not fit the capabilities of portable devices, especially of resource-limited ones. Novel middleware-level solutions are required to transparently and dynamically adapt semantic-based service provisioning to the properties of different access devices. The paper proposes a novel middleware that exploits the visibility of two kinds of metadata, user/device/service profiles and policies, to tailor the configuration of the semantic support functionalities needed to access semantic-based services, and that offers a wide set of mechanisms for making viable semantic-based service provisioning to resource-constrained portable devices.

INTRODUCTION
Advances in telecommunication and wireless systems together with the increasing diffusion of portable devices are enabling pervasive and ubiquitous service provisioning scenarios. Users expect to access the needed data from ubiquitous attachment points and even when changing physical locations, e.g., at their workplaces, at homes, at public Internet kiosks, and require context-aware services, i.e., services that can adapt provided results to changing context information, such as user relative position, user requirements, and locally available resources [3].

However, the design, development and provisioning of context-aware services in pervasive computing scenarios poses several challenges. In particular, the support of context-aware services requires appropriate models and mechanisms to gather, describe and reason about context information as users move among different environments.

Semantic languages have recently gained considerable attention within the pervasive research community as a suitable means to provide expressive context representation, querying and reasoning support [1, 2, 5]. Semantic languages permit to describe at a high level of abstraction in terms of metadata the structure and properties of the entities composing a pervasive system, e.g., users, devices and resources, and the desired management operations to govern and control entity behavior. This is especially needed in ubiquitous environments, where users move across different localities, thus experiencing a high degree of dynamicity in resource accessibility, whereas devices used to access services exhibit a high degree of heterogeneity in terms of technical properties. The adoption of Semantic Web techniques to specify and manage metadata in pervasive computing scenarios ensures that there is a common understanding between previously unknown entities about their capabilities and the current execution context and that there is no semantic gap when entities exchange context information. Moreover, Semantic Web languages enable expressive querying and automated reasoning about context representation.

However, the high degree of dynamicity and of heterogeneity of computing environments increases the complexity of exploiting Semantic Web techniques for the provisioning of semantic-based services. It is impossible to make a priori assumptions about the available users in one locality and about the semantic support functionalities, e.g., ontology repositories, inference engines and knowledge management tools, that are available on each access terminal and/or that are needed to exploit semantic-based applications.

In addition, semantic support services typically require a large amount of computational/memory resources that may not fit resource-constrained devices. Possibly strict limitations exist about the kind of semantic support facilities that can be hosted on devices. A large ontology base of, for instance, hundreds of KB is unlikely to be hosted on a PDA, which typically exhibits a memory (RAM) capacity that is significantly smaller. In addition, it is worth noting that semantic support facilities, e.g., ontology reasoning engines, tend to introduce a significant overhead that may be intolerable for portable devices. For instance, the Hp iPAQ Pocket Pc 5500, which is equipped with 128 MB of RAM, is unlikely to be able to execute a reasoning process on-board, not only because this task may be energy consuming, but also because it would probably monopolize all available memory resources, thus preventing any other application from executing on the device.

We claim that the dynamicity and the heterogeneity of pervasive computing environments require novel middleware solutions capable of adapting the semantic service support facilities to the different user requirements, to the various device properties and to the rapidly changing environment conditions. This novel middleware should be able to configure semantic support functionalities according to devices properties and user needs.

Some initial research ideas are starting to emerge that aim to exploit semantic techniques for service provisioning in pervasive environments [1, 2, 4]. However, these solutions are generally designed for specific purpose, e.g., enabling smart spaces applications or adapting content to portable devices, and provide limited support forms of configurable semantic support [11]. None of them enables to adapt the configuration of semantic support facilities to the various capabilities of portable devices.

This paper proposes a novel framework, called MASS (**M**iddleware for **A**daptive **S**emantic **S**upport). for the provisioning of configurable semantic support facilities to portable devices with possibly strict resource limitations. MASS focuses on two particular aspects. First, it exploits metadata not only to describe user/device properties, but also to describe the characteristics of semantic support services and the management choices in their provisioning. On the basis of metadata information, MASS tailors semantic support functionalities according to the various user preferences and device characteristics. In addition, MASS allows each portable device to discover and to exploit the semantic support facilities offered by other mobile devices. This is obtained through the propagation of the visibility of the semantic support functionalities provided by each device to other devices. The paper is organized as follows. Section 2 presents an overview of semantic support functionalities with respect to the specific needs of pervasive applications. Section 3 illustrates MASS metadata model and middleware architecture. Section 4 describes a prototype implementation of the proposed middleware and a preliminary usage evaluation. Concluding remarks and future research directions follow.

SEMANTIC WEB AND PERVASIVE COMPUTING

Until now, research within the pervasive community has been mainly concerned with the design and development of semantic-enabled prototype applications that exploit the benefits of a semantic approach to context awareness. However, little attention have been paid to analyse and evaluate the feasibility issues deriving from the exploitation of semantic technologies especially in pervasive computing scenarios. In order to actually exploit semantic techniques within pervasive applications, it is crucial to take into account semantic support management issues, which until now have been primarily addressed by the research community in the field of knowledge management, and reconsider them from the particular perspective of ubiquitous environments.

In particular, novel solutions are needed that are able to manage the ontological knowledge with respect to the characterizing properties of pervasive environments, i.e., the high degree of distribution in computation and resources, the wide heterogeneity of involved devices and users, and the intrinsically dynamic nature of applications behavior.

According to [10], a middleware supporting semantic technology should provide the infrastructure with the following features:

- Support for *ontology repositories* providing at least the basic storage services in a scalable, reliable and distributed fashion.
- Support for *reasoning modules* suitable for various domains and applications.
- *Multi-protocol client access* to allow different users and applications to use the system in the most appropriate and efficient way.
- *Knowledge control* in order to make possible the management of ontologies with respect to their evolution and changes.

Repository

As far as repositories are concerned, various knowledge base systems (KBS) have been developed for processing Semantic Web information, with different characteristics [9]. Some of them are memory-based, e.g., OWLJessKB [15], while others use secondary storage to provide persistence, e.g., DLDB-OWL [14], and some others support both, e.g., Sesame [13] and Jena [8]. Beside the traditional centralized model of KBS, new paradigms tend to emerge that rely on a completely distributed information model, like for instance P2P approaches for exchanging and storing ontological information [19], Semantic Web Services networks built over P2P networks [22], semantic-based topologies of P2P networks to facilitate service discovery [24] as well as Semantic Grids [23]. This kind of approaches are proposed as opposing to more traditional solutions that rely on the centralized management of ontology.

Reasoning Engines

To enable the effective use and maintenance of knowledge repositories, it is important to define some functional interfaces [10], which simplify access to the repository and hide the implementation details behind the declaration of a well-specified set of functionalities. In particular, *tell* interfaces enable to add knowledge to the repository or initiate some processing over it, *delete* interface enable to allow deletion of elements from the repository and *ask* interfaces enable to query the repository. As a drawback of this approach, a specific implementation of each interface is required for every possible support that needs to be integrated to the middleware.

A knowledge base must be associated to a reasoning engine that is able to exploit the information included in it. In particular, in [10], authors outline two typical cases in which reasoning features are needed: ontology development and ontology use. Ontology development requires reasoning tasks that can be defined as *terminological reasoning*, e.g., checking whether a class definition is consistent with respect to a given set of class descriptions, or verifying whether a class definition is more general than another. On the other hand, ontology use involves an already developed ontology and instances of data defined on that ontology, which typically consist of a higher magnitude, e.g., thousands of instances. Examples of *instance reasoning* tasks are finding all individuals in a data set that are instances of a certain class or more complex queries involving more than one class. A basic rationale behind the choice of a particular reasoner for a specific application should be the reasoning efficiency in the typical usage cases.

Depending on the application model and requirements, the appropriate reasoner must be chosen. Let us note that in pervasive applications time constraints may be very restrictive. If, for example, a user is looking for a list of nearby restaurants, the response of the semantic-enabled discovery process must be necessarily produced within a reasonable time limit, and an incomplete reasoner that is able to provide some correct answers, even if not all of them, may serve better than a complete one that requires too much time to perform its reasoning process. Moreover, as mobile devices typically experience problems of battery shortage, it is crucial that the semantic support is able to perform its task without introducing a too heavy overhead that could not be sustained by a portable device.

Another key issue is concerned with the architectural choice of whether to separate or integrate ontology and data representation from the reasoning over them. Intermediate solutions may be adopted to best fit the application needs. For example, if the reasoner and the repository are co-located on a device that can host both of them, their integration may guarantee a good level of performance, thus avoiding the useless consumption of energy due to communication overhead and the waste of memory that data duplication would imply. On the other hand, if the repository and the reasoner reside on different hosts with limited memory resources, then it would be necessary to rely on a selective transfer mechanism, which allows for the exchanging of relevant, i.e., needed for reasoning, data sets and ontologies without introducing a too heavy communication overhead. Let us note that communication consumes energy and, as we have previously pointed out, energy saving is a crucial point for the efficient management of mobile devices.

Client access

In context-aware applications, access to knowledge, as well as its representation, has to be context-dependent [10]. Due to the high dynamicity of operating context conditions that is typical of pervasive environments, it is important to assure that semantic support is accessible trough a variety of means and in different situations. This requirement holds both from a technology-centered and from a user-centered point of view, thus leading to the specification of the following features:

- Support for different transport, communication and interaction protocol - to allow various devices to access the available semantic resources. For instance, a query might be forwarded using a messaging service or via http.
- Support for customizable access and presentation of information - to avoid information overload and to present information at the right level of granularity. This can be achieved by taking into account the experience, the preferences and the needs of the user. For example, the results of semantic-enabled search might be provided on a desktop with full details, while on a palmtop the knowledge information must be restricted to the bare minimum.

Ontology management

As pervasive environments are intrinsically characterized by a high heterogeneity and dynamicity, we cannot expect related ontologies to be static. In fact, in real-world applications, ontologies are often developed by several persons and typically continue to evolve because of changes in the real world or adaptation to different tasks. According to [10], ontologies must have a *network* architecture and must be *dynamic*. Hence, ontology management for ubiquitous applications has to deal with both heterogeneity in space and heterogeneity in time. As far as the first issue is concerned, there are two possible approaches: ontologies may be merged into a single new ontology or they can be kept separated. In both cases, the different ontologies must be *aligned*, i.e., they have to be brought to a mutual agreement, by means of mapping techniques [11]. Moreover, it is essential to properly manage ontology evolution over time. Beside advanced versioning methods for the development and maintenance of ontologies [20], configuration management is needed to identify, interpret and find relations between ontology versions.

All these aspects may come together in integrated *ontology library systems* [10], which can help in the grouping and reorganizing ontologies for further reuse, integration, maintenance, mapping and versioning. In particular, we emphasize the need of management functionalities, to support the efficient maintenance of existing ontologies, and adaptation functionalities, to support the integration of different existing ontologies.

THE MASS FRAMEWORK

MASS is a framework that enables the provisioning of semantic-based services to portable devices by allowing the dynamic tailoring of semantic support services on the basis of users/devices characteristics and requirements. For example, a resource-limited device can benefit from the possibility to download a specific reasoner only when needed and/or to exploit the most appropriate reasoner available in the nearby of its current point of attachment.

Figure 1 shows MASS logical organization, which distinguishes metadata and services. Dynamic and heterogeneous sytems need an explicit description of their resources in order to make possible their management at run time. MASS adopts semantic metadata to describe users, devices and services properties and configuration management choices. MASS provides several tools and mechanisms to enable a portable device to share its knowledge base, to advertise its semantic capabilities and to discover locally available semantic support services, and to either download or remotely access needed semantic services depending on device properties and user requirements.

MASS Metadata Model

MASS adopts metadata for representing user, device and service characteristics, and management choices about the provisioning of semantic support services at a high level of abstraction, thus keeping a clean separation between semantic support logic and its configuration management. MASS adopts three kinds of metadata: *user/device profiles*, *semantic service profiles* and *configuration policies* (Figure 1a).

User/device profiles are composed of two subsections: the *preference* section and the *capabilities* section. Preferences express the desired configuration settings for access to semantic support services. For example, a user may request to be notified whenever new ontologies are advertised within the locality. Capabilities describe both the technical characteristics of the device (*technical capabilities*), e.g., the device storage space capacity, and the semantic functionalities that the device is capable of providing, in terms of the semantic support services currently hosted on the device (*semantic capabilities*). Each semantic capability includes a reference to the profile of the corresponding semantic service, i.e., to the semantic service profile.

Semantic service profiles are composed of the *service capabilities* part and the *service requirements* part. Service capabilities describe the properties of the support that a service is able to supply and they are specifically targeted to each semantic support service. For instance, a reasoner profile includes in its capabilities the kind of logic supported, e.g., logic programming or description logic, while an ontology repository profile contains a capability that quantifies the repository overall dimension. Similarly, service requirements are specialized for the particular type of semantic support service and express the technical and/or trust requirements that a device must own to be enabled to access the service. For example, a reasoner may require to be forwarded only queries encoded in RDQL or a repository may require authentication in order to allow access to its ontologies.

MASS adopts Configuration Policies to regulate access to semantic support according to specific user/device properties. Policies are high-level directives that express management choices [12]. In particular, MASS distinguishes two kinds of policies. Authorization policies define what can or cannot be done on specific target resources if certain context conditions are met. For example, the transfer of a certain amount of ontologies on a mobile device is authorized if the device storage space has a minimum, predetermined capacity and the manager in charge of ontology transmission is a trusted entity. Obligation policies specify configuration actions that 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 interested in any update of a certain ontology, she must be informed whenever a newer version of that ontology is released. Configuration policies can be static or dynamic. *Static configuration policies* define conditions that can be evaluated at configuration time, e.g., the maximum storage capacity of a device, while *dynamic configuration policies* require to test conditions at service provisioning time. For example, a policy may authorize the download of a semantic support service on a device only if the device battery level exceeds a certain threshold. Obviously, the battery level must be verified at the moment of actually start the download.

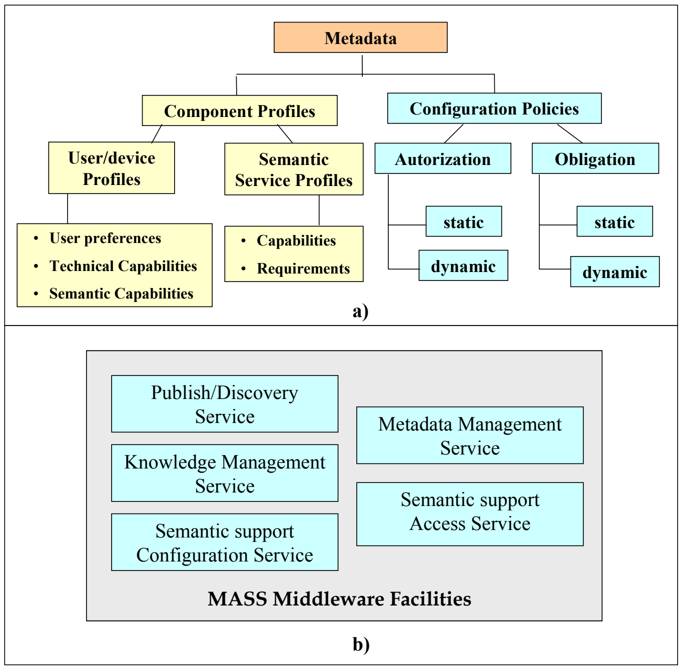

Figure 1. MASS Metadata Model and Middleware Facilities

MASS Middleware

MASS middleware services provide the needed functionalities to enable customized access to semantic support services (see Figure 1b).

The **Publish/Discovery Service** (**PDS**) allows mobile devices to advertise the semantic support services they host on their device, so that these services can be subsequently discovered and accessed by other devices. PDS receives discovery requests from mobile devices or from other middleware components that need to be provided with semantic support in order to exploit semantic-based services.

The **Knowledge Management Service** (**KMS**) is responsible for the management and maintenance of the available knowledge sources distributed among the various portable devices. This includes grouping and reorganizing ontologies on the basis of their associated profile, integrating and/or aligning related ontologies, and versioning the ones that have evolved during time.

The **Semantic support Configuration Service** (**SCS**) is responsible for adapting the semantic support services to user/device preferences and capabilities. For any user that needs to access semantic support services from her portable device, SCS parses the component profile and exploits the acquired information to take appropriate configuration decisions, depending on holding configuration policies and on currently context conditions, e.g., battery level. There are several possible configuration settings depending on the device characteristics. A portable device, such as a PDA, may be able to carry only a small amount of knowledge (or even none) and may presumably be unable to host a powerful reasoning engine. The device should then be properly configurated to exploit an external reasoner and to access needed ontologies. On the other hand, a laptop is likely to be able to perform autonomous reasoning, but may still need to update its set of ontologies to include newly released ones, in the same way as updates and recent versions of common applications are normally downloaded from the Web. On the basis of user preferences, device technical properties and configuration policies imposed in the locality, SCS selects the best configuration settings to apply to the device with respect to the various types of semantic support services. SCS identifies three configuration strategies, which can be applied to each of the three semantic support services, i.e., the reasoner, the ontology repository, or the knowledge management service.

- **Remote Support.** In this case, the user is provided with external semantic support that does not execute on her device, but on a remote host. This setting is particularly well-suited to resource-constrained devices that cannot host semantic support on-board. In particular, it is worth noting that:

- The reasoning process does not occur on-board, but it is delegated to an external reasoning engine. This option may be useful not only in case a device cannot host a reasoner, but also in case a device prefers to delegate reasoning, e.g. for saving battery, notwithstanding it could have on board the needed reasoning capabilities.

- If the reasoning process requires ontologies that are not available on user device, these ontologies are retrieved from an external repository but they are not downloaded on the device. Note that, in case reasoning is performed on-board, ontologies may be cached or stored in memory, but the storage will not persist after the completion of the ontology processing.
- Knowledge management operations are remotely performed and only the final result is provided to the user on her device. The user may provide (part of) the processable data and delegate data processing to an external component.

- **Download Support.** In this case, semantic support that is not available on user device is downloaded for subsequent usage. This setting can be applied only to those devices whose technical characteristics, e.g., memory space, and current state, e.g., battery level, allow them to download and host semantic support services.

- **Embedded Support**. In this case, the user only relies on the semantic support services that are hosted on her device, which has rich resources and management features to locally store all needed semantic support. Although this option reduces the dynamicity and variety of the provided semantic support, it may be appropriate in case the user prefers to avoid overhead and battery consumption due to communication activity, or in case of temporary disconnection from the network.

- User semantic-enabled applications can exploit only the reasoner(s) hosted by the device. This option may also result appropriate if the on-board reasoner is optimized for the particular kind of logic used within the application.
- The device hosts a local ontology repository, whose content is not altered through interaction with external knowledge sources. This self-contained approach can assure consistency to the repository, thus simplifying knowledge management operations.
- Knowledge management tools, if needed, must be directly available on the device.

The **Semantic support Access Service** (**SAS**) mediates access to semantic support services. SAS is responsible for the actual redirection of a semantic support request from a portable device to the appropriate semantic support device provider, according to the configuration choices made by SCS. SAS establishes and manages the communication session between the requesting application and the semantic support service provider, acting as a bridge that enables interoperability at both application and communication level. Interoperability at communication may be achieved either by assuring that interacting parties support at least one common protocol or by directly managing communication, thus translating incoming requests to make them compliant to the semantic service provider specifications. As far as the application level is concerned, SAS provides uniform access to semantic support services independently of the particular service instance being accessed, through a series of APIs that enable semantic-specific operations, such as retrieving an ontology or forwarding queries to a reasoning engine.

The **Metadata Management Service** (**MMS**) supports metadata specification and management, by providing various tools for metadata editing, checking for correctness, updating, removing, and browsing. In particular, MMS enables users to specify their profile, their device profile, and the profile of hosted semantic support services. Moreover, it allows the developer to specify configuration policies, which can be dynamically modified to better fit mobile users and applications needs. In addition, MMS is in charge of retrieving profiles and policies whenever they are needed by other middleware components, namely SMS and SCS.

IMPLEMENTATION DETAILS

We have developed a prototype implementation of MASS middleware, to be deployed in a wireless Internet scenario, i.e., a computing environment where wireless solutions extend the accessibility of the fixed Internet infrastructure via access points, working as bridges between fixed and mobile devices. In the following, we use the term network locality to identify a LAN with 802.11b access points as a bridge between wired and wireless devices.

As shown in Figure 2, MASS is built on top of the Java-based CARMEN system that supports the provisioning of context-dependent services to portable devices [3]. 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 [7]. 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.

In the MASS deployment scenario, mobile devices entering a locality can exploit MASS to properly access semantic support services. On the one hand, a device may need to be externally provided with semantic facilities, if it cannot or does not want to host these facilities on-board. On the other hand, portable devices with on-board semantic facilities may desire to act as servers and offer these facilities to other devices that are currently connected to the same network.

Prototype implementation

MASS is designed to integrate with third-party semantic support systems, i.e., ontology repositories, reasoning engines and knowledge management tools. These supports are often provided as integrated components, like for instance ontology repositories that support basic reasoning facilities, such as Sesame, DLDB-OWL and Jena, or ontology repositories that support knowledge management tools, e.g., SHOE [http://xml.coverpages.org/shoe.html] offering ontology versioning. Autonomous components providing semantic support services are also available. This is the case of most reasoners, such as the description logic inference engines OWLJessKB [15], FaCT [17] and Racer [18], and the Java Theorem Prover [16], which supports first order logic but also includes special purpose reasoners. An example of autonomous knowledge management tool is the web-based system OntoView [20].

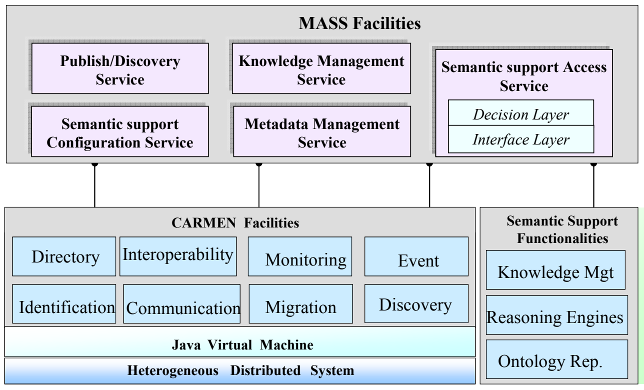
MASS Facilities
Publish/Discovery Service
Knowledge Management Service
Semantic support Access Service
Decision Layer
Interface Layer
Semantic support Configuration Service
Metadata Management Service
CARMEN Facilities
Directory
Interoperability
Monitoring
Event
Identification
Communication
Migration
Discovery
Java Virtual Machine
Heterogeneous Distributed System
Semantic Support Functionalities
Knowledge Mgt
Reasoning Engines
Ontology Rep.

Figure 2. MASS Prototype Middleware

The current prototype implementation interoperates with semantic support integrated systems, e.g., Sesame and DLDB-OWL. These systems have been chosen considering the good level of performance they have exhibited in first benchmarking experiments [9]. As far as metadata specification is concerned, MASS exploits Web Ontology Language (OWL) [6] to express both profiles and policies. In particular, to express policies, MASS adopts the KAoS ontology [6]. An excerpt of user/device profile is shown in Figure 3. To describe device characteristics like available storage space, we rely on the hardware ontology developed at UMBC within the CoBrA framework [1].

The MASS middleware initiates a configuration process whenever a new user enters a CARMEN domain for the first time and wishes to be provided with semantic support. Profiles and policies, along with the reasoning abilities needed to interpret them, are used by the MASS middleware during the configuration phase. In particular, when a user first enters the domain, her profile and her device's profile are supplied to MASS. The information acquired through profile parsing and reasoning is exploited, together with the rules encoded in the configuration policies, to take appropriate decisions about the configuration of access to semantic support facilities.

The Publish/Discovery Service is implemented as a registry that publishes semantic services profiles for subsequent discovery and retrieval. Users willing to share their semantic support services with co-located mobile users can advertise the services via PDS by publishing their profile. Users that need to be provided with semantic support can query PDS to find out if a service is locally available that fulfills their request.

The Knowledge Management Service is designed to coordinate with PDS to organize, classify and semantically characterize the advertised semantic services. Our current implementation of KMS is able to parse and reason about service profiles, e.g., to compare the capabilities listed in the profiles of two reasoning engines. KMS can perform its tasks on demand, i.e., whenever it receives an explicit request from a user or from PDS, or it can automatically execute tasks at the occurrence of certain events, e.g., every time a new repository is advertised. Although KMS is equipped with a dedicated set of knowledge management tools, it may also rely on tools provided by mobile devices, which are retrieved via PDS.

```
<rdf:Description rdf:about="#UserA">
  <profile:OntologyStrategyPreference
         rdf:resource="#Embedded"/>
  <profile:OntologySourcePreference
         rdf:resource="#Web"/>
  <profile:ReasoningStrategyPreference  rdf:resource="#Embedded"/>
         ...
```

```
<rdf:RDF
   xmlns:dev  "http://daml.umbc.edu/ontologies/cobra/0.4/device#"
   xmlns:xsd  "http://www.w3.org/2001/XMLSchema#"
>
<dev:DeviceMemory rdf:ID="20GBMemory">
   <dev:amountOfMemory
         rdf:datatype="&xsd;nonNegativeInteger">20</amountOfMemory>
<dev:memorySizeUnit

         rdf:datatype="&xsd;string">Gigabyte</memorySizeUnit>

<rdf:Description rdf:about="#LaptopA">
  <profile:techCapability
       rdf:resource="#20GBMemory"/>
  <profile:semanticCapability
         rdf:resource="#JTPReasoner"/>
  <profile:semanticCapability
         rdf:resource="#PhotoOntology"/>

</rdf:Description>  ...
</rdf:RDF>
```

Figure 3. MASS Example Profile

The Semantic support Configuration Service component acquires the profile of each user that enters the locality and parses it to determine user and device technical characteristics. On the basis of profile information and currently holding configuration policies, SCS takes appropriate configuration decisions. For instance, let us consider a configuration policy that imposes to adopt the remote strategy for ontology provisioning if the device overall storage space capacity does not reach a minimum threshold. Let us suppose that a user states in her profile that she would prefer to download updated ontologies on her device. SCS will evaluate both the policy and the profile and, if the device storage space dimension results to be under the predefined threshold, it will set a remote support strategy for ontology provisioning. The selected settings are then collected and stored by SCS within a user configuration profile that is maintained in a dedicated repository, the Configuration Settings Repository. In addition, if the user device hosts a semantic support service on-board that the user wishes to locally advertise, SCS automatically coordinates with PDS to publish this service.

The Semantic support Access Service mediates user access to semantic resources by both exploiting the information recorded within the configuration profile and by enforcing the rules encoded in dynamic configuration policies. SAS is designed in two layers, the Decision Layer (SAS-DL) and the Interface Layer (SAS-IL). In particular, the SAS-DL component reads the configuration profile to determine which type of support strategy, i.e., remote, download or embedded, has been chosen for the required support by SCS. In case semantic support must be externally provided to the user application, i.e., in case a remote or download strategy has been set by SCS, SAS-DL interacts with PDS to discover a candidate semantic service that fulfils the user application request. If dynamic configuration policies hold, SAS-DL is responsible for retrieving applicable policies and for testing their activating conditions to determine which actions has to be taken. For instance, a policy may allow the download of semantic support on a device only if the device battery level storage space exceeds a certain value. In this case, SAS-DL component will test the device battery level before allowing any support download. Once an appropriate semantic support service has been found, SAS-DL coordinates with SAS-IL in order to actually enable access to the semantic service functionalities. SAS-IL represents the interface between the middleware and the semantic support services supplied by third parties. Therefore, it provides a set of APIs that allow some basic semantic-specific operations such as adding/removing knowledge to/from repositories, forwarding queries to reasoners and asking for support to knowledge management elements. SAS-DL forwards the portable device request to the selected semantic support service provider by exploiting the APIs offered by SAS-IL. Let us note that the APIs are independent from the particular semantic support component being accessed, in order to grant portability and interoperability between MASS middleware and various externally provided semantic services. Therefore, whenever a new semantic support needs to be integrated to the middleware, a specific implementation of SAS-IL must be implemented. At present we have implemented a SAS-IL version for Sesame.

Finally, the Metadata Management Service is currently implemented as a graphical user interface that permits the specification of user/device profile and semantic support service profile according to the underlying semantic OWL profile ontologies. Another graphical tool is provided to specify static and dynamic configuration policies. Profiles and policies are stored in a LDAP-compliant Metadata Repository managed by MMS, which is in charge of retrieving them when needed by PDS.

MASS at Work

We have tested MASS in the design and implementation of a prototype semantic-based discovery application, which retrieves services available in the vicinity on the basis of semantic requirements specified by the user. Our testbed setting consists of a wireless network composed by several 802.11 network localities, with each locality modeled as a CARMEN domain, as depicted in Figure 4. Each domain provides execution environments (*places*) for shadow proxies on each physical node, offers MASS middleware facilities and hosts third-party semantic support service providers. Resources, entities and services are described by means of semantic metadata. In this application prototype, each place hosts an instance of PDS, SCS, SAS and MMS, whereas KMS is implemented as a centralized element that reside only on one predefined place within the domain. In addition to this, each place is provided with a configuration

settings repository and a metadata repository to store user/device profiles and configuration policies. MASS service functionalities are accessible from wireless devices through several access points and users interact with the infrastructure via device-specific clients running on their wireless access devices.
The clients allow users to subscribe to the MASS-based semantic discovery application, by filling in a form provided by MMS with user/device and semantic support service profile, and to authenticate themselves to the service before starting any semantic discovery session. When a user first accesses the service, MASS instantiates a shadow proxy in the domain where the user is currently attached.

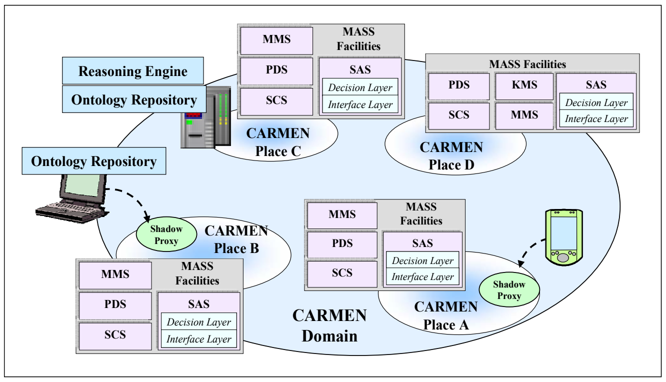

Figure 4. MASS Deployment Scenario

Let us examine how MASS services interact by considering the case of a laptop that is equipped with an ontology repository on-board, but hosts no reasoner. The user specifies her profile via the MMS local facility. In particular, in the preference section of the profile, the user states that she needs to access an external reasoner and that she prefers to use the repository hosted on her laptop. In case new ontologies are needed to deduce inference, she would like to download and store them in the laptop repository. In addition, she sets the Web as the default source for downloading ontologies. In the semantic capabilities part of the profile, a semantic support service is listed, i.e., the ontology repository. After the specification phase, SCS initiates the configuration of the laptop, by parsing the information included in user/device profile. In particular, following user preferences, SCS sets the remote support strategy and the embedded strategy for, respectively, reasoning and ontology service provisioning. Then, it sets the download strategy for ontology provisioning in case needed ontologies are not included in the laptop repository and indicates the Web as the first-choice source from which ontologies should be retrieved. In case the preferred source is not available, SCS decides that ontologies will be searched among the repositories hosted by portable devices in the locality. These configuration settings are recorded within a configuration profile that is stored at the current place. Subsequently, SCS sends to local PDS registry the profile of the semantic support service hosted by the laptop, i.e., the repository, so that PDS can publish it.

Let us now suppose that the laptop user is looking for a local service providing digital photo printing. To this extent, the user wishes to exploit the semantic-enabled discovery service that is available on her laptop to find an appropriate service. The semantic discovery application needs semantic facilities to execute a match between the functionalities required by the user, i.e., high quality photo printing, and the functionalities offered by locally available services, as expressed in the service profiles [21]. Let us note that this task requires both a reasoner and the ontologies that are used to express service properties within service profile.

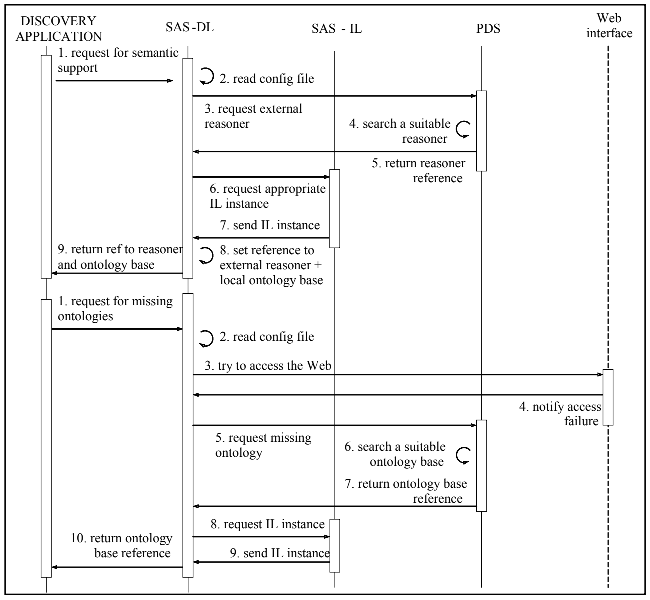
DISCOVERY APPLICATION
SAS-DL
SAS - IL
PDS
Web interface
1. request for semantic support
2. read config file
3. request external reasoner
4. search a suitable reasoner
5. return reasoner reference
6. request appropriate IL instance
7. send IL instance
9. return ref to reasoner and ontology base
8. set reference to external reasoner + local ontology base
1. request for missing ontologies
2. read config file
3. try to access the Web
4. notify access failure
5. request missing ontology
6. search a suitable ontology base
7. return ontology base reference
8. request IL instance
10. return ontology base reference
9. send IL instance

Figure 5. Case study Interaction Flow

Therefore, the semantic-enabled discovery application interacts with MASS to obtain proper access to semantic support services, namely to a reasoner and to needed ontologies. The interaction flow between the discovery application and the middleware components is shown in Figure 5.
Let us now suppose that, during discovery, the reasoner occurs to be in need of some ontologies that are not available at the on-board repository. Hence, the reasoner requests to MASS to be provided with missing ontologies, according to the configuration settings stored within the configuration profile (see Figure 5).

CONCLUSIONS AND FUTURE WORK

Semantic languages have recently gained attention as a means of expressing context-related metadata in pervasive computing applications. However, the exploitation of semantic support requires a considerable amount of memory and computational resources that may not fit resource-limited devices. We propose a novel middleware which is capable of adapting semantic support to the different characteristics of mobile devices and provides mobile users with the visibility on semantic functionalities hosted by nearby devices.

Although the proposed middleware provides mobile devices with enhanced capabilities to exploit heavyweight semantic support, it is responsible for introducing some overhead that may degrade system performance. This overhead is partly due to the initial configuration phase and partly caused by the complexity of the semantic matching algorithms that are used to compare semantic support requests and offers. We are analyzing the effect of such overhead by evaluating performance parameters of the prototype discovery application, such as delay time for a discovery request.

We are currently testing MASS prototype in different scenarios to evaluate its applicability and usefulness. Current and future work is primarily concentrating on the analysis of various available semantic support services and tools for subsequent integration with our prototype middleware. We also believe that in such an open and dynamic scenario it will be necessary to deal with security issues. Therefore, we are planning to enhance the MASS framework with security features, e.g., access control features.

REFERENCES

[1] Chen, H., et al.: Semantic Web in a Pervasive Context-Aware Architecture. In: Artificial Intelligence in Mobile Systems 2003, Universitat des Saarlandes, Saarbrucken, Germany (2003).

[2] Masuoka, et al.: Task Computing – the Semantic Web meets Pervasive Computing. In: Proc. of ISWC 2003, Sanibel Island, Florida, October 2003.

[3] Bellavista, et al.: Context-Aware Middleware for Resource Management in the Wireless Internet. IEEE Transactions on Software Engineering, 29 - 12 (2003).

[4] Agostini, A., et al.: Integrated Profile and Policy Management for Mobile-Oriented Internet Services. Technical Report Firb-Web-Minds N. TR-WEBMINDS-04.

[5] Wang, X., et al.: Semantic Space: An Infrastructure for Smart Spaces. IEEE Pervasive Computing, Vol. 3, No. 3, July-September 2004.

[6] Uszok, A., et al.: KAoS policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement. In: Proc. of POLICY 2003, 4-6 June 2003.

[7] Fuggetta, et al.: Understanding Code Mobility, IEEE Transactions on Software Engineering, Vol. 24, Issue 8 (1998).

[8] The Jena SW Framework, http://jena.sourceforge.net/.

[9] Guo, Y., et al.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. In: Proc. of ISWC 2004, Springer-Verlag, Heidelberg, LNCS 3298, 2004.

[10] Davies, et al. (eds.): Towards the Semantic Web, John Wiley &Sons, England, 2003.

[11] Omelayenko, B.: RDFT: A Mapping Meta-Ontology for Web Service Integration, In: Omelayenko, B., Klein, M. (eds.), Knowledge Transformation for the Semantic Web, IOS Press (2005).

[12] Corradi, A., et al.: Policy-driven Management of Mobile Agent Systems, In: Proc. of POLICY 2001, Bristol, UK. Springer-Verlag, LNCS 1995 (2001).

[13] Broekstra, J., et al.: Sesame: a Generic Architecture for Storing and Querying RDF and RDF Schemas. In: Proc. of ISWC 2002, Sardinia, Italy. LNCS 2342, Springer-Verlag (2002) 54-68.

[14] Pan, Z., et al.: DLDB: Extending Relational Databases to Support Semantic Web Queries. In: Workshop on Practical and Scalable Semantic Systems, ISWC 2003, Sanibel Island, Florida, USA, October. LNCS 2870, Springer-Verlag, Berlin (2003).

[15] Kopena, et al.: DAMLJessKB: A Tool for Reasoning with the Semantic Web. In: Proc. of ISWC 2003, Sanibel Island, Florida, USA, October. LNCS 2870, Springer Verlag, Berlin (2003).

[16] Fikes, R., et al.: JTP: A System Architecture and Component Library for Hybrid Reasoning. In: Proc. of SCI 2003, Orlando, Florida, USA (2003).

[17] Fact. http://www.cs.man.ac.uk/~horrocks/FaCT/

[18] Haarslev V., et al.: Racer: An OWL Reasoning Agent for the Semantic Web. In: Proc. of WSS '04, in conj. with WI 2004, IEEE Computer Society Press.

[19] Tummarello, G., et al.: RDFGrowth, a P2P annotation exchange algorithm for scalable Semantic Web applications. In: Proc. of P2PKM 2004, in conj. with Mobiquitous 2004, Boston, Massachusetts, USA (2004).

[20] Klein, M., et al.: Ontoview: Comparing and versioning ontologies. In: Collected Posters ISWC 2002, Sardinia, Italy (2002).

[21] Toninelli, A., et al.: Semantic Discovery for Context-Aware Service Provisioning in Mobile Environments. In: Proc. of MCMP Workshop, in conj. with MDM 2005, Ayia Napa, Cyprus, May 9-13 (2005).

[22] Schlosser, et al.: A scalable and ontology-based P2P infrastructure for Semantic Web Services. In: Proc. of P2P 2002, Linkoeping, Sweden, (2002).

[23] Cannataro, M., et al.: Semantics and knowledge grids: building the next-generation grid. In: IEEE Intelligent Systems, 19 (1), Jan-Feb 2004 Page(s):56 – 63.

[24] Hao Ding, et al.: A vision on semantic retrieval in P2P network. In: Proc. of AINA 2004, Fukuoka, Japan, March 29-31 (2004).