

Adaptive Semantic Middleware for Mobile Environments

Antonio Corradi, Rebecca Montanari, Alessandra Toninelli

Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna, Italy

Email: {acorradi, rmontanari, atoninelli}@deis.unibo.it

Abstract—Context-awareness is considered a key driving principle for the design and provisioning of adaptable pervasive services. Rightfully describing and interpreting context, however, is a challenging issue. Semantic technologies are emerging as effective means to describe and reason about context information and to allow unknown entities to have a common understanding of context. However, the exploitation of semantic technologies for the design/deployment of context-aware applications in pervasive environments replete with heterogeneous devices requires to address several issues. In particular, a crucial aspect is how to support semantic-based service provisioning to mobile devices with limited capabilities. Novel solutions are required to transparently and dynamically adapt semantic-based service provisioning to the properties of different access devices. The paper proposes a middleware-level solution approach that exploits the visibility of two kinds of metadata (profiles and policies) to support the configurability of the semantic support functionalities depending on user/device properties, and that offers a wide set of mechanisms for making viable semantic-based service provisioning even to resource-constrained portable devices.

Index Terms—Semantic Web, pervasive computing, mobile computing, configuration middleware, adaptive services.

I. INTRODUCTION

Advances in telecommunication and wireless systems together with the increasing diffusion of portable devices are enabling a pervasive service provisioning scenario where users can access needed data/services from ubiquitous attachment points and even when changing physical locations, e.g., at their workplaces, at homes, at public Internet kiosks. The new arising paradigm of enabling service provisioning anywhere, anytime, and to any device demands appropriate models and mechanisms to design and deploy context-aware services, i.e., services that can adapt provided results to changing context information, such as user relative position, user requirements, locally available resources, and device characteristics [1]. Supporting context-awareness is a complex management issue as it involves the consideration of several new challenges. In particular, the design, development and deployment 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 suitable means to provide expressive context representation, querying and reasoning support [2][3][4]. Semantic languages permit to describe at a high level of abstraction in terms of metadata the structure and properties of the users, devices and resources composing a pervasive system (profiles), and the desired management operations to govern and control entity's behavior (policies). This is especially needed in pervasive computing scenarios where users move across different environments which they cannot have a priori knowledge of, experience unpredictability and dynamicity in resource availability and access services from heterogeneous devices. In addition, the adoption of semantic technologies to specify and manage metadata ensures that there is a common understanding among unknown entities about user/device/resource capabilities and that there is no semantic gap when entities exchange context information. Moreover, semantic languages enable expressive querying and automated reasoning about context representation.

However, despite the promising features of semantic technologies for pervasive computing, the high degree of heterogeneity and dynamicity of pervasive environments complicate the uptake of semantic-based context-aware services in fielded systems. Mobile devices have limited processing power, memory, and battery capabilities, and they are typically unsuited for accessing traditional semantic-based services designed for fixed networks. Semantic support services, e.g., ontology repositories, inference engines and knowledge management tools, typically require a large amount of computational/memory resources that may not fit the properties of devices. In particular, strict limitations exist about the kind of semantic support facilities that can be hosted on resource-constrained 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. It is worth noting that also 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 semantic-driven reasoning process about context on-board, not only because this task may be too energy consuming, but also because it would probably monopolize all available

This paper is based on "Dynamic Configuration of Semantic-Based Service Provisioning to Portable Devices," by A. Corradi, R. Montanari, and A. Toninelli, which appeared in the Proceedings of the IEEE International Symposium on Applications and the Internet (SAINT), Phoenix, USA, January 2006. © 2006 IEEE.

memory resources, thus preventing any other application from executing on the device.

In addition, mobile devices usually move among environments in unpredictable way, without static knowledge about the locally available resources and services. For instance, it is impossible for a resource-limited device to rely on pre-fixed availability of semantic support functionalities in its point of attachment. This requires dynamic discovery of semantic support facilities provided by both available co-located devices and hosting environment, dynamic configuration and set-up of retrieved support services accordingly to device's characteristics and binding to locally available semantic infrastructure.

Some initial research ideas are starting to emerge that aim to exploit semantic techniques for context-aware service provisioning in pervasive environments and to define appropriate architectural support [2][3][5]. However, these solutions are generally designed for specific purpose, e.g., enabling smart spaces applications or adapting content to portable devices, and provide limited level of configurability of semantic support facilities depending on the various device's properties. We claim that the heterogeneity and dynamicity of pervasive computing environments require novel middleware solutions capable of adapting the configuration and set-up of semantic service support facilities to the different user requirements, to the various device properties and to the rapidly changing environment conditions.

This paper proposes a novel framework, called MASS (Middleware for Adaptive Semantic Support) for the provisioning of configurable semantic support facilities to portable devices with possibly strict resource limitations. MASS exploits metadata to describe not only user/device properties, but also the characteristics/properties of semantic support services and the configuration management preferences. MASS allows each portable device to discover needed locally available semantic support facilities through the propagation of the visibility of the semantic support functionalities provided by co-located devices. On the basis of metadata information, MASS tailors semantic support functionalities according to the various user preferences and device characteristics.

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, and Section 4 describes a prototype implementation of the proposed middleware. Section 5 presents and evaluates the functioning of MASS in a discovery application scenario. Finally, concluding remarks and future research directions follow.

II. SEMANTIC WEB AND PERVASIVE COMPUTING

Great research efforts have been recently spent for the design/development of semantic frameworks for context-aware service provisioning. The solution proposals in the literature tend to address two main issues: semantic languages for context representation and semantic support

frameworks.

Among the various models proposed to represent context information [6] [7], ontologies with well defined declarative semantics have proved to be the most suitable model by enabling knowledge sharing in open, dynamic systems, by allowing efficient reasoning on context information and by providing the means for networked services to collaborate in a non-ambiguous manner. Recent approaches to ontology-based context modeling are SOUPA [8] and CONON [9] that define two-level ontologies for context modeling. A core ontology defines generic concepts that are usually modeled in context such as platform or users, while more specific ontologies introduce concepts for a particular application domain such as characteristics of users and devices, and location, or the definition of intelligent environments such as home or office. More generic, in [10] a simple context ontology is defined which is easily extensible and allows the description of semantic Web services. Interoperability is also studied in [11], which proposes an approach for mapping concepts between different ontologies.

Several semantic frameworks have been also developed to shift the intelligence of context building, storage and decision making to the middleware level, thus enabling for greater flexibility in implementation [2] [3]. Semantic middleware infrastructures typically provide support for ontology repositories providing scalable and reliable storage services, for reasoning modules suitable for various domains and applications, for multi-protocol client access to allow different users and applications to use the system in the most appropriate and efficient way, and for knowledge control in order to make possible the management of ontologies with respect to their evolution and changes.

The main weakness of all above ontology proposals and semantic infrastructures is that they require complex and heavyweight support infrastructures that can be deployed typically only over traditional Internet-based networks with powerful, highly available and reliable devices. On the contrary, technological advances in hand-held devices are stressing the need to incorporate semantic technologies and support facilities also into portable devices with different capabilities and network connectivity (both wired and wireless) in order to get the benefits of easier and more flexible ubiquitous service access. Embedding semantic technologies into portable devices gives rise to several management issues that require to rethink traditional semantic infrastructures in order to take into account the key characteristics 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 service behavior and execution environment. Up to now, despite the increasing interest, little attention has been paid to build the kind of semantic support infrastructure needed to exploit semantic technologies for context-aware service provisioning to portable devices. To provide reasoning support on a mobile device, for instance, the current common solution is to connect to a special reasoning server on

the fixed network. However, cellular Internet connections are quite expensive, unreliable, and not always available. In addition, if many mobile devices access simultaneously the same reasoning server, scalability issues may arise.

Probably due to the number of complex technological challenges related to the design of semantic facilities for mobile devices, few solutions in the literature already address the problem of semantic-based context-aware service provisioning to portable devices [12]. This section overviews for each semantic support facility state-of-the-art solutions for both fixed and mobile networks by examining their features and by identifying the novel design guidelines to follow in order to design and deploy semantic support facilities that take into account the characteristics of pervasive environments and that can provide context-aware service provisioning to heterogeneous mobile devices.

A. Repository

Various ontology repositories have been developed for processing and storing Semantic Web information over traditional Internet network [13]. Some of them are memory-based, e.g., OWLJessKB [14], others use secondary storage to provide persistence, e.g., DLDB-OWL [15], and some others support both, e.g., Sesame [16] and Jena [17]. Most current-state-of-the-art ontology repositories share the common feature to provide centralized knowledge base systems (KBS) model. Centralization, however, is restrictive in many ways for mobile pervasive computing environments: it creates performance bottlenecks and induces significant administrative overhead when applied to heterogeneous and dynamic networks.

New solutions based on distributed models are recently emerging for exchanging and storing ontological information. In particular, P2P computing offers the promise of removing many problems posed by traditional centralized solutions. P2P allows a highly decentralized repository network with peer nodes exploiting ontologies present at other nodes of the network without intervention of a central server. Emerging P2P solutions, such as Semantic Web Services networks built over P2P networks [18], semantic-based topologies of P2P networks to facilitate service discovery [19] as well as Semantic Grids [20], seem suited to address the characteristics of pervasive environments in presence of high mobility and heterogeneity of users/devices.

However, several challenges arise in integrating P2P models and ontologies. Merging (composing) ontologies from multiple autonomous sources is critical for the pervasive computing environment. In particular, it is necessary to incorporate descriptions of new classes of entities (devices, services, components, and so on) and new types of context information (e.g., new sensors) as they are introduced. In addition, successful research results in merging ontologies must be implemented in standard services and libraries, in order to be integrated into the infrastructure. To enable the effective use and maintenance of knowledge repositories, it is important

to define interoperable functional interfaces [saint 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.

B. Reasoning Engines

Reasoning engines are needed to effectively exploit the information stored into knowledge bases. In particular, reasoning is crucial into two cases [saint 10]: ontology development and ontology use. Ontology development requires reasoning support to check whether a class definition is consistent with respect to a given set of class descriptions, or to verify whether a class definition is more general than another (terminological reasoning). 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. Ontology use requires reasoning to find all individuals in a data set that are instances of a certain class or to check the consistency of an instance data with respect to the ontology (instance reasoning). Several autonomous reasoners are already available for traditional networks, each one tailored to a specific application model and requirements, such as the description logic inference engines OWLJessKB [14], FaCT [21], Pellet [22] and Racer [23], and the Java Theorem Prover [24], which supports first order logic but also includes special purpose reasoners. The typical deployment scenario is to have ontologies and reasoners tightly coupled into the same resource-rich device.

There are several issues in order to design and implement reasoners for dynamic pervasive computing environments with heterogeneous, often resource-constrained, devices. A major issue is to correlate information provided by distributed devices to furnish a more comprehensive view of the context they habit. A meaningful correlation of heterogeneous data collected from different networked sources is achieved through reasoning and consists in exploiting relations among data in order to provide a more comprehensive and informative view on the set of significant properties characterizing the environment. This correlation task can be achieved by endowing the devices with the suitable inferential power, but embedding reasoning engines into devices rises several challenges due to the strict limitations posed by device's resources. Mobile devices may not have the possibility to host onboard reasoners due to CPU and memory space limitations. Also battery constraints should be taken into account: it is crucial that reasoning performs without introducing a too heavy overhead that could not be sustained by a portable device with battery problems. In addition, due to limited network connectivity time constraints may be very restrictive. If, for example, a user is looking for a list of nearby restaurants, the response of the reasoning 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.

Another key issue to take into account 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. Communication consumes energy and, as we previously pointed out, energy saving is a crucial point in networks with mobile devices. To the best of our knowledge, the most relevant work that tries to enable reasoning for mobile devices is the KRHyper system developed specifically for Java-enabled mobile devices that supports reasoning onboard mobile devices [12].

C. Client Access

A crucial requirement in pervasive environments is to ensure accessibility to semantic support facilities through a variety of means and in different situations. To address this requirement client access facilities should be developed following proper design guidelines. Client access facilities support different transport, communication and interaction protocols 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 is also needed 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.

D. 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 [25], 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 [26]. Moreover, it is essential to properly manage ontology evolution over time. Beside advanced versioning methods for the development and maintenance of ontologies [27], configuration management is needed to identify, interpret and find relations between ontology versions.

All these aspects may come together in integrated ontology library systems [25], 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 of adaptation functionalities to support the integration of different existing ontologies.

III. THE MASS FRAMEWORK

MASS framework enables semantic-based service provisioning to portable devices by allowing the dynamic configuration of semantic support services on the basis of user/device characteristics and requirements. Figure 1 shows MASS logical organization, which distinguishes metadata and services. Dynamic and heterogeneous systems need an explicit description of their resources in order to facilitate their run-time management (discovery, access and operation). MASS adopts semantic metadata to describe user/application/device/service 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 choose whether to download or remotely access needed semantic services depending on device properties and user requirements.

A. MASS Metadata

The MASS reference model for configuration identifies client applications, devices, and semantic services as the key entities involved in configuration processes. In particular, MASS considers any software component providing a semantic support facility as a service, i.e., a "black box" encapsulating physical/logical resources and providing the functions to operate on them. Client applications on mobile devices request to access semantic support services from heterogeneous devices.

As shown in Figure 1a, MASS adopts two kinds of metadata for representing the characteristics of client applications, devices and semantic services, and the configuration management choices about the provisioning of semantic support services: application/device/service profiles and configuration policies.

Profiles have a modular structure comprising different

parts, each one grouping metadata with a common logical meaning. In particular, they include three common key parts: identification, requirements, and capabilities. The identification part provides naming information to identify user applications/devices/semantic services. The requirements part expresses entity desiderata. Application requirements describe the semantic support functionalities needed by the application to achieve its tasks, such as reasoning features required to make inferences about a given knowledge base. Device requirements specify technical conditions that must hold for the device to properly interact with semantic services. Semantic service requirements describe conditions that client applications/devices should satisfy to access such services. For example, a reasoner may require to be forwarded only queries encoded in RDQL [28] and an ontology repository may require user authentication in order to be accessed. By focusing on the third part, user application capabilities include user attributes, such as security certificates, and application properties, such as supported query interfaces, e.g., RDQL; device capabilities represent technical characteristics, supported functions and resource state, such as memory storage capability, secure socket layer support, and battery level; semantic service capabilities describe provided support facilities and how they are achieved. For instance, a reasoner profile may include as a capability its implemented reasoning logics, e.g., Description Logic (DL) or Frame-Logic, while an ontology repository capability can describe the repository data size.

Configuration policies are high-level directives that regulate how to access semantic support services according to specific application/device properties and to configuration management [29]. In particular, MASS distinguishes two kinds of policies. Authorization policies define which actions can or cannot be performed on specific semantic support services if certain conditions are met. For example, the transfer of an ontology set on a mobile device is authorized only if the device memory size exceeds a minimum threshold. Obligation policies specify configuration actions that have to be carried out at certain event

occurrence, given that specific conditions are verified. For instance, on any update of a specific ontology, interested users should be notified about the newly released ontology version. Configuration policies can be static or dynamic. Static configuration policies define conditions that can be evaluated at configuration time, e.g., conditions on device memory size, while dynamic configuration policies require to test conditions at service provisioning time, such as in the case of a condition on battery level.

MASS configuration policies can be defined with a variable degree of granularity and with different scopes. Device policies are defined on the basis of the technical characteristics and/or the state of a single client device, such as memory size or processor speed. Application policies allow the customization of access to semantic support services depending on the client application needs and operating settings. Conversely, environment configuration policies do not focus on a single device or application, but describe management choices about the environment where user applications execute. For instance, a local configuration policy may prevent access to semantic services if the local network is congested. In addition, policies can be static or dynamic.

B. MASS Middleware

The MASS middleware components facilitate profile and policy encoding, support advertising and retrieval of semantic support services, configure and manage access to semantic services on behalf of user applications.

The **Metadata Management Service (MMS)** allows the specification, modification, checking for correctness, parsing, and installation of application/device/semantic service profiles and of configuration policies. MASS adopts description logics to model and reason about user/device/service metadata by representing profiles as ontologies [30]. Conversely, configuration policies are expressed as rules according to the following pattern:

```
if condition then config_setting
```

that corresponds to a Horn clause, where predicates in the head and in the body are represented by classes and properties defined in the profile ontologies.

The **Publish/Discovery Service (PDS)** supports advertising and retrieval of semantic support services hosted on board of user devices. PDS allows users to advertise the semantic support functionalities hosted on their mobile and/or fixed devices by publishing appropriate semantic service profiles. In addition, PDS enables mobile users that need semantic support functionalities to properly run their semantic-based applications to discover needed semantic services. In particular, PDS supports the discovery of semantic support services by performing semantic matchmaking between user application/device/semantic service requirements and capabilities. The semantic matchmaking process allows to identify semantically compatible services, i.e., services whose capabilities are semantically similar with user/device requirements and whose requirements are semantically sim-

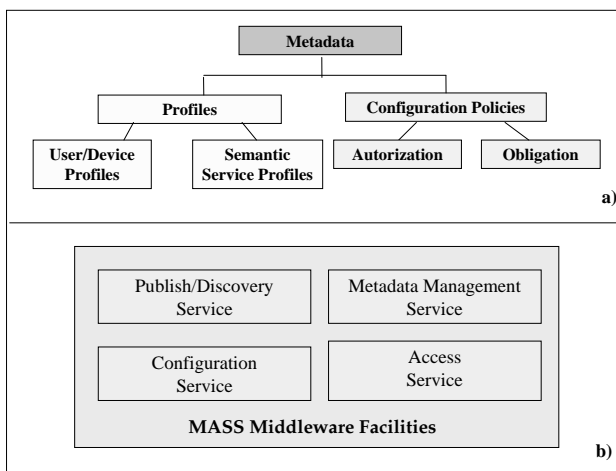


Figure 1. MASS metadata and middleware components.

ilar with user/device capabilities.

The **Configuration Service (CS)** is responsible for providing user applications with needed semantic support functionalities, depending on application/device/service properties and defined configuration policies.

There are several possible configuration alternatives. A portable device such a smart phone may be able to carry only a small set of ontologies and may be unable to host a powerful reasoning engine. Therefore, the device should be properly configured 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. In addition, in case of absent or difficult Internet connection, it would be more advantageous for a device to rely on its on-board semantic facilities in order to avoid useless battery consumption while trying to access semantic services hosted on remote devices. CS 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 Strategy.** 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. not to cause overload of its processing resources, 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 Strategy.** 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 Strategy.** 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 **Access Service(AS)** is responsible for the actual binding of user applications to needed semantic support services, according to the established configuration settings. To retrieve needed semantic service instances, either locally or remotely depending on the configured strategy, the AS component interacts with the PDS component. In addition, AS provides an application programming interface (API) that offers to client applications standard methods, such as OWL-API, for accessing semantic services and translates these methods to low-level calls to specific semantic service instances. Depending on the deployment environment, different ways to communicate with semantic support services may be desirable, from HTTP in a Web environment to RMI, to SOAP in a Web service context. In order to allow the highest flexibility, AS provides protocol handlers as intermediaries between semantic services and their clients, each one handling a specific communication protocol.

IV. IMPLEMENTATION DETAILS

We have developed a prototype implementation of MASS framework, to be deployed over a wireless Internet network, i.e., a network environment where access points extend the accessibility of the fixed Internet infrastructure via by working as bridges between fixed and mobile devices. In particular, we define network locality a LAN with 802.11b access points as a bridge between wired and wireless devices.

In the considered network deployment scenario, mobile devices entering a locality can exploit MASS to properly access semantic support services. Different access possibilities are available. On the one hand, a device may need external semantic facilities, if it cannot or does not want to host these facilities on-board. On the other hand, a device with on-board semantic facilities may desire to act a server and to offer these facilities to other devices

that are currently connected to the same network.

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 [1]. The CARMEN facilities provides mechanisms and tools to address most common issues in context-aware service provisioning to wireless clients. In particular, MASS exploits CARMEN naming system to identify services/resources, the CARMEN context management service that includes a monitoring facility to observe indicators at both the application and at the system level, and an event management facility to be notified about relevant indicator variations to interested entities, and a low-level discovery service that supports searching and retrieval of node/service within the device locality.

A. Prototype Implementation

The Metadata Management Service provides graphic tools for the specification and management of application/device/service profiles and policies. As Figure 3a-b show, MASS adopts OWL-based formats for profile representation[31]: each profile element specifies the logical profile part it belongs to. Policies are represented in SWRL [32], using concepts defined in the OWL-based profile ontologies. MMS is built upon the Java-based ontology editor and knowledge base framework Protege. Among the alternative solutions for profile and policy representation available in the literature, we have chosen the standard Semantic Web language OWL because of its expressivity and SWRL as a rule language for its good level of integration with OWL [33].

The Publish/Discovery Service discovers semantic support services available in the user/device locality and semantically compares their profiles with user requirements to provide users with needed semantic support service instances. In addition, PDS allows users to advertise and publish the profile of the semantic support services hosted on their device. As far as the discovery of semantic support facilities is concerned, PDS exploits the CARMEN discovery service to find available nodes/services in the network locality. PDS builds the list of semantic support services hosted on each node. It is possible to configure CARMEN discovery service to consider different diameters for service discovery scopes. For instance,

if the diameter is 1, PDS retrieves the list of nodes available in the user network locality; if diameter is 3, all nodes within 3-hop distance from requesting clients are considered. Once obtained the list of available nodes, PDS exploits a semantic matching algorithm to discard services semantically incompatible with the requirements expressed in the application profile and the configuration settings. PDS supports semantic matchmaking according to the algorithm detailed in Section IV-B.

The Configuration Service component reads user application/device profiles to determine which semantic service capabilities applications need to perform its tasks and coordinates with CARMEN context manager to retrieve relevant information about the application and device state. On the basis of profile information and currently active configuration policies, CS takes appropriate configuration decisions. For example, consider the case of a user application that needs a DL-reasoner supporting DIG-API interface. A configuration policy may state that remote reasoning is allowed only in case the device battery level exceeds a minimum threshold, while in case of battery shortage the local reasoner must be used. Depending on the battery level, CS will set the remote or the embedded (local) strategy for the reasoning service in a configuration file dedicated to store desired configuration settings. 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 addition, CS may coordinate with It is possible to instruct CS to perform configuration at application start-up or on explicit user request.

The Access Service is responsible for the actual binding of user applications to semantic services, according to the configuration profile settings. AS consists of two layers, the Decision Layer (AS-DL) and the Interface Layer (AS-IL). AS-DL coordinates interaction with the other middleware components and manages access to semantic services instances according to the required configuration. AS-IL represents the interface between the middleware and the semantic support services supplied by third parties and implements sets of standard APIs, e.g., DIG-API and OWL-API, that allow methods such as adding/removing knowledge to/from repositories and forwarding queries to reasoners. AS-IL translates the invocation of these methods into calls to specific semantic services. The current prototype provides the access interface implementation to access the Jena framework and the Pellet DL-based inference engine. These semantic support systems have been chosen because they both support OWL-API-based access methods.

At user application start-up, the AS-DL component reads the configuration profile to determine which support strategy has been set for the required semantic service. In order to be provided with suitable semantic service instances according to the client application profile, AS-DL interacts with PDS. In case external semantic support is needed according to a remote or download strategy, PDS discovers appropriate service instances among the

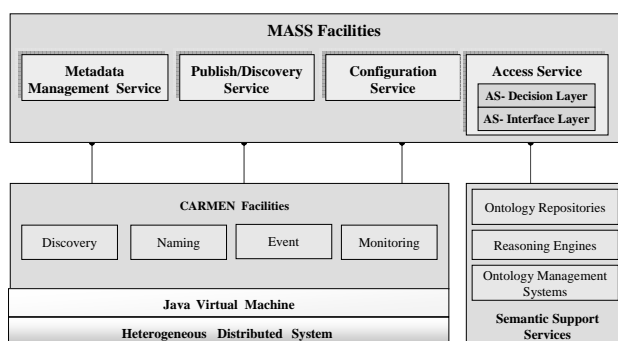


Figure 2. The MASS layered architecture

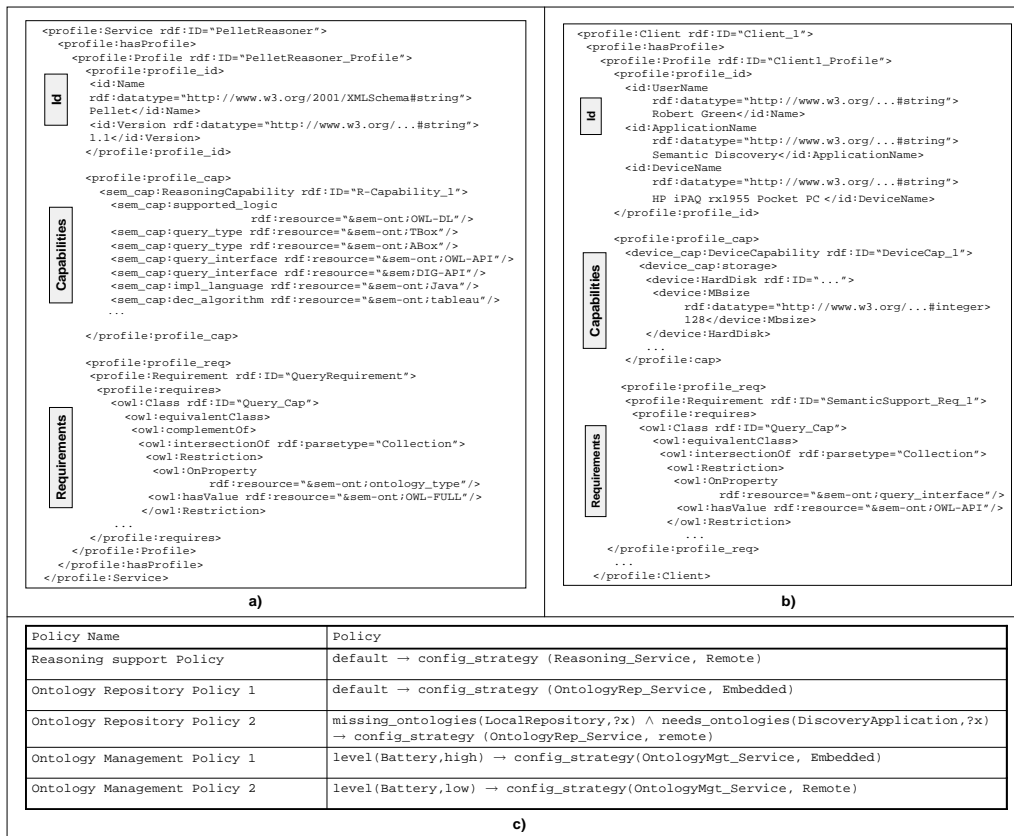


Figure 3. The MASS semantic support service/user profiles and configuration policies.

semantic services available in the network vicinity. On the contrary, in case of embedded strategy, PDS redirects AS-DL to the semantic service instances hosted on the user device. Once an appropriate service instance is found, SAS-DL binds the appropriate AS-IL implementation to the client application, so that any invocation of the API methods is translated into a call to the specific semantic service.

B. Matching Algorithm

In this section we provide some insights on the matching algorithm performed by PDS. Our algorithm, shown in Figure 4, can identify semantically compatible services, i.e., whose capabilities semantically match with user application requirements and whose requirements semantically match with user application capabilities.

Each capability is characterized by a set of properties. For example, the capability of providing a reasoning service is characterized by properties such as the supported logic, the query language and the implementation language. In order to exploit the inference abilities of a reasoning engine, it is necessary to properly encode capabilities and requirements in OWL. Hence, following a Description Logic-based model, offered capabilities are represented as individuals, i.e., specific instances of a capability class, whereas requirements are represented as requested capability classes. In particular, request classes are defined by restrictions, i.e., imposing

certain limitations to the values that their characterizing properties can assume. For instance, if one is looking for an ontology management system supporting OWL-API queries, then the corresponding request class is defined by restricting the values of the query language property to OWL-API.

In particular, the algorithm takes an offered capability and a required capability as inputs and returns the degree of semantic compatibility between them. The algorithm recognizes three kinds of semantic similarity relationships: the offered capability may be an instance of the requested requirement class (exact case), or an instance of a more generic requirement class (subsumes case), or an instance of a more specialized requirement class (plug-in case). For example, let us consider a requested capability being the ability for an ontology management system to support persistent ontology storage. If a system exhibits exactly the capability to support persistent storage, then we have a case of exact match. If a system offers instead the more generic capability of supporting both persistent and volatile ontology storage, we have a case of subsumes match. Finally, if a system offers the capability of providing database storage, which is a particular kind of persistent storage, we have a plug-in match case.

Figure 4 shows in detail the steps of the matching algorithm. The current prototype is implemented in Java. It exploits the semantic web framework Jena [17] to

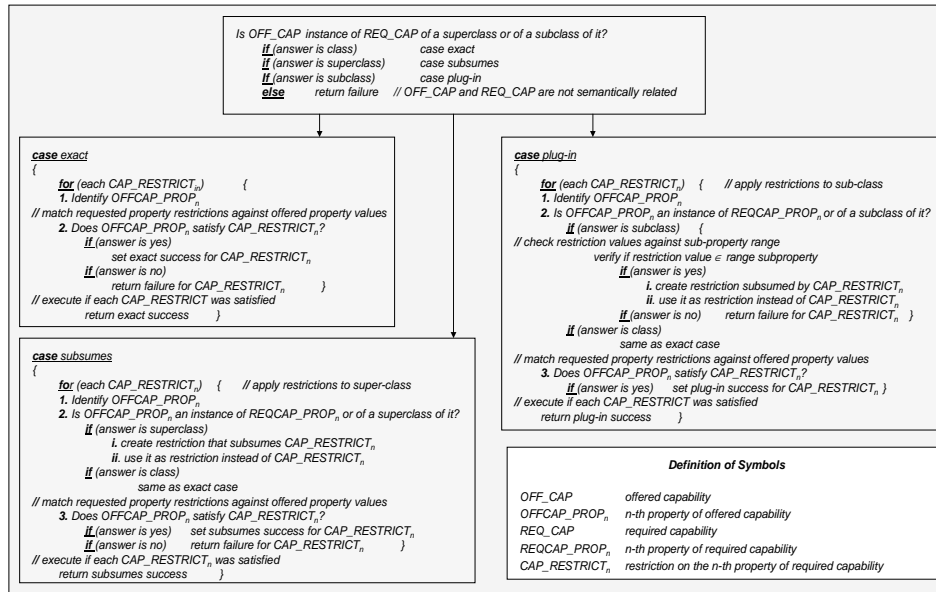


Figure 4. The PDS semantic matching algorithm.

store ontologies and extract knowledge from them, and the OWL-DL reasoner Pellet [22] to perform inference on given ontologies, e.g., recognizing subclass and superclass relations.

V. CASE STUDY

We have tested MASS in the design and implementation of a prototype semantic-based discovery application, which retrieves services providing tourist information available in the user proximity on the basis of semantic requirements specified by the user. Our test-bed setting consists of both a fixed computing infrastructure and mobile devices connected via IEEE 802.11 Access Points (APs). Each AP defines a network locality that includes local wired hosts and mobile wireless devices, playing the role of service clients or providers, within the AP coverage area. For instance, one AP locality may include the Pellet reasoning service, whose profile is depicted in Figure 3a. At MASS deployment time, it is necessary to decide where to allocate middleware components. In the above scenario, we have decided to distribute all CARMEN low-level facilities on fixed hosts in the network: there is one centralized directory for service/node registration, while one instance of any other CARMEN middleware component is allocated on fixed hosts in each AP locality. About MASS high-level facilities deployment, mobile devices have all MASS middleware components installed. In particular, MASS generates on each mobile device a local instance of MMS, PDS, CS and AS. It is worth noting that the PDS matching algorithm is implemented as a tiny J2ME application that relies on a relatively small set of ontologies (about 200 KB) and a lightweight reasoner [12], in order to be able to execute on mobile devices.

Let us examine how MASS services interact by considering the case of the discovery application running

on a PDA. Semantic discovery requires reasoning capabilities to make inferences on the semantic metadata describing users and services [34]. In particular, user requirements and service capabilities are modeled by means of semantic-based profiles, which must be matched in order to discover in the network locality proper services to fulfil a user request. This means that the application needs to be provided with an ontology repository and management tool to store and manipulate ontologies, with ontologies describing user and service profiles, and with a reasoning engine to make inferences about ontologies. The reasoner and the ontology base are accessed via OWL-API. The needed semantic functionalities are specified by the discovery application provider in the user application profile by means of MMS, shown in Figure 3b.

Let us suppose that the PDA is equipped with the ontology management framework Jena [17] hosting the ontologies describing only user requirements, but not service capabilities. Let us also suppose that the PDA hosts no generic reasoner on-board. The user exploits MMS to specify her configuration policies, which are shown in Figure 3c. In particular, she sets the remote strategy for the reasoning service. As far as ontologies are concerned, the user wishes to exploit the ontologies hosted on her device (embedded strategy) and to be provided with external service-specific ontologies without permanently downloading them (remote strategy). The user also sets the embedded strategy for the ontology repository and management service, unless the battery level is low; in this case, an external ontology management system must be used, in order not to waste battery due to communication between the external reasoner and the local ontology base.

After the specification phase, CS initiates the configuration process. Let us consider the case of the PDA having full battery. In this case, CS sets the remote strategy for

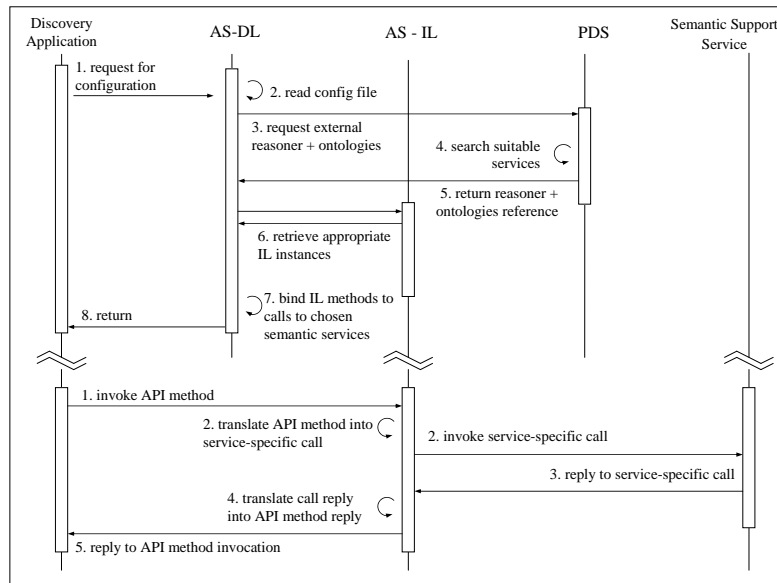


Figure 5. Case study interaction flow

the reasoner and for needed ontologies, and the embedded strategy for the ontology management system.

Now the AS component looks for needed semantic service instances. In particular, from the user application profile AS-DL retrieves the local ontology management framework Jena and the local ontologies. To discover an appropriate reasoner and the source of needed ontologies, it coordinates with PDS. The corresponding interaction flow is depicted in Figure 5. Let us suppose that a Pellet reasoner instance, whose profile is shown in Figure 3a, resides on a mobile device connected to the same Access Point locality as the PDA, while missing ontologies are available on a fixed node. After performing its semantic matching algorithm between the user requirements and the Pellet reasoner capabilities, PDS forwards to AS-DL a reference to the Pellet instance and to the ontology source nodes. AS-DL instantiates a Pellet-compliant AS-IL implementation of the reasoner access interface, and a Jena-compliant AS-IL implementation of the ontology base access interface and performs appropriate binding to redirect the OWL-APIs invoked by the semantic discovery application to the chosen semantic support services, as shown in Figure 5.

The exploitation of a middleware for semantic service provisioning, such as MASS, introduces different forms of overhead, depending on both the deployment environment conditions and the performance of the different middleware facilities. Configuration of semantic support and access management to semantic service represent the most time and resource consuming activities. A non-negligible performance degradation may derive from the initial configuration process. In particular, additional time is required at application start-up for semantic-based reasoning and matching whenever external semantic support services must be provided. However, we claim that PDS enhanced discovery flexibility and precision in retrieving

semantic support services largely counterbalance the response time degradation. It is also worth noting that the number of available semantic support tools is rather limited because of their high specialization. Hence, the PDS component generally performs the semantic matching algorithm over a limited set of services, thus making the time degradation deriving from semantic-based reasoning acceptable.

Another significant performance issue may arise when a download or remote strategy must be enforced. While the first case causes an initial peak of resource consumption, especially bandwidth and battery ones, due to the download of a semantic support service on the mobile device, the latter case might increase resource usage at constant rate but over the whole application lifetime, because of data exchange between the application and the remote semantic service. A quantitative evaluation of the overhead introduced by the configuration process is highly dependent from both deployment setting conditions, e.g., local bandwidth and network traffic, and application-specific parameters, e.g., ontology size and reasoning complexity. Let us note that the expressivity and flexibility of our framework allows to define configuration policies according to different strategies. These strategies can be tailored to best adapt the provisioning of semantic support services to different operating conditions and application-specific parameters, making the adoption of semantic functionalities viable for mobile devices.

VI. CONCLUSION

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.

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.

ACKNOWLEDGMENT

This work is partly supported by MURST PRIN Project "MOMA: a middleware approach to MOBILE Multimodal web services".

REFERENCES

- [1] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli, "Context-aware middleware for resource management in the wireless internet." *IEEE Trans. Software Eng.*, vol. 29, no. 12, pp. 1086–1099, 2003.
- [2] H. Chen, T. Finin, and A. Joshi, "Semantic Web in in the Context Broker Architecture," in *Proceedings of the Second Annual IEEE International Conference on Pervasive Computer and Communications*. IEEE Computer Society, March 2004.
- [3] R. Masuoka, B. Parsia, and Y. Labrou, "Task computing—the semantic web meets pervasive computing," 2003.
- [4] X. Wang, D. Zhang, J. S. Dong, C. Chin, and S. R. Hettiarachchi, "Semantic space: A semantic web infrastructure for smart spaces," *IEEE Pervasive Computing*, vol. 3, no. 3, pp. 32–39, July–September 2004.
- [5] A. Agostini, C. Bettini, and D. Riboni, "Loosely coupling ontological reasoning with an efficient middleware for context-awareness," in *Proceedings of MobiQuitous 2005. The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*. IEEE Computer Society Press, 2005, pp. 175–182.
- [6] B. N. Schilit, N. Adams, M. M. T. Rich Gold, and R. Want, "The parctab mobile computing system," in *Proceedings of the Workshop on Workstation Operating Systems*, 1993, pp. 34–39.
- [7] M. Strimpakou, I. Roussaki, C. Pils, M. Angermann, P. Robertson, and M. E. Anagnostou, "Context modelling and management in ambient-aware pervasive environments," in *Proceedings of the Workshop on Location and Context Awareness (LoCA)*, ser. Lecture Notes in Computer Science, T. Strang and C. Linnhoff-Popien, Eds., no. 3479. Springer Verlag, 2005, pp. 2–15.
- [8] H. Chen, F. Perich, T. Finin, and A. Joshi, "Soupa: Standard ontology for ubiquitous and pervasive applications," in *Proceedings of MobiQuitous 2004. The International Conference on Mobile and Ubiquitous Systems: Networking and Services*, I. C. S. Press, Ed., 2004, pp. 258–267.
- [9] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, "Ontology based context modeling and reasoning using owl," in *Proceedings of the PerCom Workshops*, 2004, pp. 18–22.
- [10] D. Preuveneers, J. V. den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, and K. D. Bosschere, "Towards an extensible context ontology for ambient intelligence," in *Proceedings of EUSAI*, ser. Lecture Notes in Computer Science, P. Markopoulos, B. Eggen, E. H. L. Aarts, and J. L. Crowley, Eds., vol. 3295. Springer Verlag, 2004, pp. 148–159.
- [11] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt, "C-owl: Contextualizing ontologies," in *Proceedings of the International Semantic Web Conference*.
- [12] A. Sinner and T. Kleemann, "Krhypcr in your pocket," in *Proceedings of the Conference on Automated Deduction*, ser. Lecture Notes in Artificial Intelligence, no. 3632. Springer Verlag, 2005, pp. 452–457.
- [13] Y. Guo, Z. Pan, and J. Heflin, "An evaluation of knowledge base systems for large owl datasets." in *International Semantic Web Conference*, ser. Lecture Notes in Computer Science, S. A. McIlraith, D. Plexousakis, and F. van Harmelen, Eds., vol. 3298. Springer, 2004, pp. 274–288.
- [14] J. Kopena and W. Regli, "Damljesskb: A tool for reasoning with the semantic web," in *Proceedings of the International Semantic Web Conference*, ser. Lecture Notes in Computer Science, no. 2870. Springer Verlag, October 2003.
- [15] Z. Pan and J. Heflin, "Dldb: Extending relational databases to support semantic web queries," in *Workshop on Practical and Scalable Semantic Web Systems, ISWC 2003*, pp. 109–113. [Online]. Available: <http://www.cse.lehigh.edu/heflin/pubs/psss03-poster.pdf>
- [16] J. Broekstra, A. Kampman, and F. van Harmelen, "Sesame: A generic architecture for storing and querying rdf and rdf schema," in *Proceedings of the First International Semantic Web Conference*, ser. Lecture Notes in Computer Science, I. Horrocks and J. Hendler, Eds., no. 2342. Springer Verlag, July 2002, pp. 54–68.
- [17] "Jena: A semantic web framework for java." [Online]. Available: <http://jena.sourceforge.net/>
- [18] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, "A scalable and ontology-based p2p infrastructure for semantic web services," in *Proceedings of P2P 2002*, 2002.
- [19] H. Ding, I. Solvberg, and Y. Lin, "A vision on semantic retrieval in p2p network," in *Proceedings of AINA*, march 2004.
- [20] M. Cannataro and D. Talia, "Semantic and knowledge grids: Building the next-generation grid," *Intelligent Systems (ISSI-0095-1203) - Special Issue on E-Science*, vol. 19, no. 1, pp. 56–63.
- [21] "The fact reasoner." [Online]. Available: <http://www.cs.man.ac.uk/horrocks/FaCT/>
- [22] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: a practical owl-dl reasoner," *Submitted for publication to Journal of Web Semantics*. [Online]. Available: <http://www.mindswap.org/papers/PelletJWS.pdf>
- [23] V. Haarslev, "Racer: An owl reasoning agent for the semantic web," in *Proceedings of WSS '04, in conj. with the Web Intelligence Conference (WI)*. IEEE Computer Society Press, 2004.
- [24] R. Fikes, "Jtp: A system architecture and component library for hybrid reasoning," in *Proceedings of SCI 2003*, 2003.
- [25] J. Davies, D. Fensel, and F. van Harmelen, Eds., *Towards the Semantic Web*. Wiley, 2002.
- [26] B. Omelayenko, "Rdfit: A mapping meta-ontology for web service integration." in *Knowledge Transformation for the Semantic Web*, 2003, pp. 137–153.
- [27] M. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov, "Ontoview: Comparing and versioning ontologies," in *Collected Posters ISWC 2002, Sardinia, Italy, June 2002*. [Online]. Available: <http://www.cs.vu.nl/mcklein/papers/ISWC02-poster.pdf>

- [28] A. Seaborne, "Rdql - a query language for rdf, w3c member submission 9 january 2004." [Online]. Available: <http://www.w3.org/Submission/RDQL/>
- [29] A. Corradi, N. Dulay, R. Montanari, and C. Stefanelli, "Policy-driven management of agent systems." in *POLICY*, ser. Lecture Notes in Computer Science, M. Sloman, J. Lobo, and E. Lupu, Eds., vol. 1995. Springer, 2001, pp. 214–229.
- [30] D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, *The Description Logic Handbook*, F. Baader, Ed. Cambridge University Press, 2003.
- [31] "Owl web ontology language reference, w3c recommendation 10 february 2004."
- [32] "Swrl: A semantic web rule language combining owl and ruleml, w3c member submission 21 may 2004."
- [33] B. Parsia, V. Kolovski, and E. Sirin, "Extending shoiq(d) with dl-safe rules: First results," in *Proceedings of the International Workshop on Description Logic Workshop (DL)*, 2006. [Online]. Available: <http://mindswap.org/kolovski/dlsafe-DL2006.pdf>
- [34] P. Bellavista, A. Corradi, R. Montanari, and A. Toninelli, "Context-aware semantic discovery for next generation mobile systems," *IEEE Communications Magazine*, vol. 44, no. 9, p. to be published, 2006.

Antonio Corradi graduated from University of Bologna, Italy, and received MS in electrical engineering from Cornell University, USA. He is a full professor of computer engineering at the University of Bologna. His research interests include distributed and parallel systems and solutions, middleware for pervasive and heterogeneous computing, infrastructure support for context-aware multimodal services, network management, mobile agent platforms. He is member of IEEE, ACM, and AICA.

Rebecca Montanari graduated from University of Bologna, Italy, where she received PhD degree in computer science engineering in 2001. She is now an associate professor of computer engineering at the University of Bologna. Her research primarily focuses on policy-based networking and systems/service management, mobile agent systems, security management mechanisms, and tools in both traditional and mobile systems. She is member of IEEE and AICA.

Alessandra Toninelli graduated from University of Bologna, Italy, where she is currently a PhD student in computer science engineering. Her research interests focus on semantic-based middleware supports for service provisioning, context-aware services, security solutions for pervasive environments, policy-based service management and mobile agent systems. She is member of IEEE and ACM.