Semantic web technologies for ubiquitous computing resource management in smart spaces

John Soldatos*, Kostas Stamatis,
Siamak Azodolmolky, Ippokratis Pandis
and Lazaros Polymenakos

Athens Information Technology
19, 5km Markopoulo Avenue
P.O. Box 68, GR-19002 Peania, Greece
E-mail: jsol@ait.edu.gr
E-mail: ksta@ait.edu.gr
E-mail: sazo@ait.edu.gr
E-mail: ipan@ait.edu.gr
E-mail: lcp@ait.edu.gr
*Corresponding author

Abstract: Context-aware ubiquitous computing environments tend to be highly distributed and heterogeneous, while also featuring increased dynamism as elements, devices and middleware components join, leave and change their status. In such environments, information is derived and fused with numerous sensors and context-aware middleware components. As a result, directory and naming services, along with reasoning mechanisms, are at the heart of any non-trivial ubiquitous computing application. In this paper, we argue that semantic web technologies can deal with directory service requirements of ubiquitous computing environments, much more efficiently than the wide range of legacy mechanisms. To justify this claim, we introduce a model that could greatly facilitate the development, deployment and management of ubiquitous computing applications. This model relies on semantic web technologies (*i.e.*, ontology management) and facilitates the integration of hardware and middleware elements in the scope of a ubiquitous computing application. Using this model and its underlying ontology management schemes, we implemented proof-of-concept applications in the scope of a smart space comprising numerous sensors, actuators and middleware components. Based on the implementation experience, we outline the merits of using semantic web technologies in ubiquitous context-aware computing and smart spaces.

Keywords: web ontologies; semantic web; OWL; Ubiquitous Computing (Ubicomp); context-awareness.

Reference to this paper should be made as follows: Soldatos, J., Stamatis, K., Azodolmolky, S., Pandis, I. and Polymenakos, L. (xxxx) 'Semantic web technologies for ubiquitous computing resource management in smart spaces', *Int. J. Web Engineering and Technology*, Vol. X, No. Y, pp.000–000.

Biographical notes: John K. Soldatos obtained his Dipl-Eng degree in 1996 and his PhD in 2000, both from the the National Technical University of Athens. He has had an active role in numerous EC funded (ESPRIT, ACTS, IST, FP6) research projects. He has also consulted in many ICT projects for leading enterprises. He has co-authored more than 80 papers published in

international journals and conference proceedings. Since March 2003, he is with AIT, where he is currently an Assistant Professor. His current research interests are in network control and management, grid, pervasive and autonomic computing.

Konstantinos Stamatis obtained his Master of Engineering degree in 2002 from the National Technical University of Athens and his Master of Science degree in Athens Information Techology, after the completion of MSIN program of Carnegie Mellon University. Mr. Stamatis has knowledge of C, C++ and Java programming experience in implementing software applications, as well as in Unified Modeling Language. His current research interests lie in the fields of hardware design and ubiquitous/pervasive computing systems. He is actively involved in the CHIL-FP6-506909 project.

Siamak Azodolmolky received his computer hardware (BEng) degree from Tehran university in Iran in 1994 and his first master degree (MEng) in computer architecture from Azad University, in 1998. He has worked with the Data Processing Iran (ex-IBM), Tehran, Iran since 1992 as Systems Engineer and Senior R&D Engineer during 1992–2001. He received his second MSc degree with distinction from the Information Networking Institute of the Carnegie Mellon University. His research interests include performance evaluation of computer networks and autonomic and ubiquitous computing. In 2006, he joined Athens Information Technology as a researcher, while also pursuing a PhD.

Ippokratis Pandis is a PhD candidate at Carnegie Mellon University (CMU). Before joining CMU, he was member of the Autonomic and Grid Computing research group of Athens Information Technology (AIT), where he worked for the CHILFP6-506909 project. Pandis holds publications to several conferences in the areas of database systems, middleware for ubiquitous computing and hypertext/hypermedia systems. Ippokratis got his BSc from the Computer Engineering and Informatics Department (CEID) of the University of Patras, Greece, and his MSc from the Information Networking Institute (INI) of the CMU.

Lazaros C. Polymenakos obtained his Electrical Engineering and Computer Science degree from the National Technical University of Athens Greece (1989), and his Masters (1991) and Doctoral degrees (1995) from the Massachusetts Institute of Technology, Cambridge, MA, USA. Since 1995, he has worked with the IBM Human Language Technologies. In 1996, he was Visiting Professor at Rutgers University, NJ, USA. In 1998, he joined IBM Hellas, SA. Since 2002, he has been a faculty member of Athens Information Technology, where he focuses on distributed systems, multimodal applications and signal processing. He is the author of numerous publications, presentations and patents.

1 Introduction

The vision of Ubiquitous Computing (Ubicomp) paradigm aims at exploiting the full range of sensors, devices and networks available to transparently provide services, regardless of time and the location of the end user (Weiser, 1991). Ubicomp services are essentially context-aware services, since they acquire and process information about their surrounding environment. Context-aware applications execute service logic not only based on input provided explicitly by end users, but also based on implicitly derived information (Dey *et al.*, 2001). There are several approaches to derive implicit information, for example:

- Tag-based approaches, where tags are read to track objects and infer context (*e.g.*, Want *et al.*, 1992).
- The wearable computing paradigm, where sensors and customised input-output operations are invoked on equipment which is put on humans (see for example Smailagic and Siewiorek, 2002).
- Smart spaces that use sensors and effectors to achieve natural interaction between the humans and the environment (and vice versa) (*e.g.*, Johanson *et al.*, 2002).
- Specialised wired environments, which connect numerous devices that interact and interoperate.

Sophisticated ubiquitous computing applications are likely to leverage more than one of these approaches to capture context, leading to hybrid schemes. For instance, wearable computing applications and reading of tags may operate in the scope of smart spaces.

The majority of the context acquisition approaches outlined above acquires implicit information based on a rich set of casually accessible, often invisible sensors that are connected to a network structure. Sensors provide information streams, which are accordingly processed by middleware components towards deriving context cues. As a characteristic example, in the scope of smart spaces, complex perceptive interfaces and recognition algorithms process audiovisual streams and extract context relating to the identity and location of people and objects. Having an initial set of context cues, information fusion algorithms are applied to derive composite context, leading to situation recognition. Context-acquisition can then trigger the service logic of ubiquitous computing services. The latter is likely to be composed of a rich set of invocations to soft-computing services, including commands to actuating devices, sensor control and regulation of the environment. Therefore, a ubiquitous context-aware computing infrastructure includes:

- a transparent sensing infrastructure, which is as non-intrusive as possible
- infrastructures for controlling sensors and actuating devices
- a collection of perceptual components gaining elementary context cues from the sensor streams
- information fusion components combining elementary context cues to more sophisticated context.

The above families of hardware, software and middleware components can be seen as the common denominator of any non-trivial ubiquitous computing application. These components are characterised by extreme diversity in terms of functionality, underlying technologies and vendors. Heterogeneity and diversity are therefore inherent in ubiquitous computing and context-aware applications. Moreover, ubiquitous computing environments are characterised by increased dynamism in the sense that sensors, devices, computing resources, and services are likely to dynamically join or leave (Yau *et al.*, 2002). Managing heterogeneity and dynamism is therefore crucial to facilitating application development. Dealing with diversity and dynamic environments hinges on directory services that constantly maintain and provide information about the components comprising such large-scale complex distributed systems. Such a directory service should provide information about all hardware, software, and semantic middleware components, so that the latter can dynamically be registered, discovered, and invoked at runtime.

Apart from directory services, sophisticated ubiquitous computing applications are likely to comprise reasoning middleware. The reason is that these applications must, in several cases, take autonomous decisions towards delivering ambient intelligence services. Reasoning is for example required in the following cases:

- To infer or correlate heterogeneous components or entities. The later may relate either to physical objects (*e.g.*, people, objects, artefacts, sensors) or even to software and middleware components (*e.g.*, tracking components, recognition algorithms).
- To resolve uncertainty, when conflicting information is fused from multiple sources. As an example, different instances of recognition algorithms may provide conflicting information on the identity of a person.
- To automatically infer and apply rules from other existing rules, relating to user preferences, models or even behavioural patterns.

Reasoning relies on knowledge about the surrounding environment and is therefore closely related to directory services.

Numerous middleware solutions for directory services have been developed over the recent years (Balazinska *et al.*, 2002; Czerwinski *et al.*, 1999; Zhu *et al.*, 2003; Guttman *et al.*, 1999).[1–4] These solutions are tailored to specific tasks, such as locating services in the scope of a Service-Oriented Architecture (SOA)[4] or achieving services and devices interoperability.[2] However, none of these mechanisms are appropriate for an effective representation of entities (*i.e.*, physical world entities and software/middleware components) in the scope of a ubiquitous computing environment. Moreover, they do not provide support for reasoning. Thus, these mechanisms cannot essentially facilitate the development, integration and deployment of non-trivial ubiquitous computing services.

Recently, semantic web technologies (*e.g.*, RDF(S),[5] DAML+OIL,[6] OWL[7]) have been proposed towards providing formalised standards-based conceptualisation of all kinds of knowledge. These technologies have been used in a variety of domains as manifested from the numerous ontologies that have been developed. At the same time, ontological knowledge management and representation is greatly facilitated through the emergence of tools. Prominent examples are knowledge acquisition systems like Protégé,[8] ontology management systems like Semantic Network Base[9] and Jena,[10] as well as inference engines for answering queries like RACER (Möller and Haarslev, 2006).

This paper argues that semantic web technologies are appropriate for the supporting directory services, registration mechanisms and intelligent reasoning in the scope of ubiquitous computing applications. Semantic web technologies can capture the full range of concepts engaged in a complex ubiquitous computing environment. Furthermore, they can provide support for reasoning. In justifying these claims, we developed an agent-based framework that supports the development of ubiquitous computing services, which uses an ontology management system as a knowledge management back end. The ontology management system is used to provide directory services with respect to hardware, software and middleware elements, as well as context information.

Hardware (*e.g.*, sensors) and middleware elements (*e.g.*, recognition algorithms) register their information with an ontological knowledge base. Following this registration, other components can interact with the ontology to dynamically discover, invoke or manage those elements. Thus, the ontology management system allows components (such as sensors, actuators and context-aware elements) to be integrated in a ubiquitous computing environments through minimal effort (*i.e.*, in an almost *plug n' play* fashion). The registration model based on an ontological knowledge base can therefore be used to support a deployment model, where infrastructure providers register components and service providers interact with them dynamically. This model is thoroughly described in this paper.

Apart from maintaining up-to-date information on the status of hardware and middleware components, the proposed knowledge management system keeps track of contextual information (*e.g.*, people, objects and their locations). While the ontology management systems can support any kind of knowledge, the particular instantiation considered in this paper concerns a smart space comprising cameras, microphones and a range of middleware components, which are operating within a room. We conveniently call this room *smart room*, and use it as a testbed for the development of realistic context-aware ubiquitous computing applications.

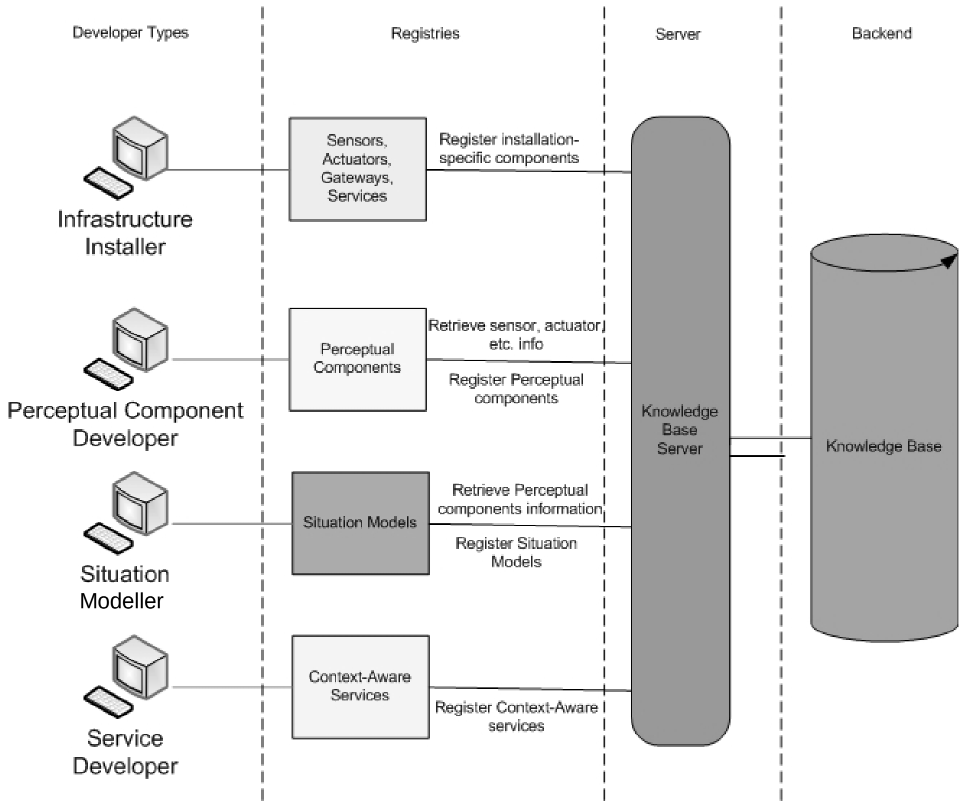
Overall, the directory services provided by the knowledge base facilitate the development and deployment of ubiquitous computing services. Based on these services, we have implemented real-life tools and ubiquitous computing applications. One of these applications, namely the Smart Space Resource Manager, constitutes a facility for monitoring and controlling sensors, perceptual components and actuating devices. Another application, namely, the memory jog, constitutes a non-intrusive service providing memory aids and other forms of assistance in meetings, lectures and conferences.

The rest of the paper is structured as follows: Section 2 introduces a model for flexible and cost-effective deployment of ubiquitous computing services. Section 3 illustrates why and how semantic web technologies, ontology management systems and a knowledge base server are at the heart of this framework. Moreover, it describes the structure and content of a proof-of-concept ontology that can support the introduced deployment model. Section 4 presents the implementation of prototype applications and middleware infrastructures which, based on the introduced deployment model, exploit the knowledge base facilities. Moreover, it discusses the merits of semantic web technologies, as the latter were manifested in the implementations. Finally, Section 5 concludes the paper.

2 Deployment model for ubiquitous computing services

Ubicomp services in smart spaces comprise a wide range of heterogeneous hardware, software and middleware components. The latter are likely to be developed and deployed by different entities-players. In order to facilitate the development and deployment of services in smart spaces, we introduce the deployment model depicted in Figure 1. This model clearly distinguishes the players involved in the process of developing and deploying Ubicomp services, as well as the components that they provide and deploy. At the heart of this model lies a system for ontological knowledge management, conveniently called knowledge base. This knowledge base serves as a repository, in which all component providers register specific information about the characteristics and the operation of their systems. Providers can access the knowledge base to acquire information on other components. This information is exploited in installing, configuring and operating the components that comprise the ubiquitous computing service. Note that the repository can provide information that can be inferred from other pieces of data within the repository. Thus, it is overall characterised by a degree of *intelligence*, which is more thoroughly explained in the following paragraphs.

Figure 1 Deployment model for Ubicomp services in smart spaces


Developer Types
Registries
Server
Backend
Infrastructure Installer
Sensors, Actuators, Gateways, Services
Register installation-specific components
Perceptual Component Developer
Perceptual Components
Retrieve sensor, actuator, etc. info
Register Perceptual components
Situation Modeller
Situation Models
Retrieve Perceptual components information
Register Situation Models
Service Developer
Context-Aware Services
Register Context-Aware services
Knowledge Base Server
Knowledge Base

The model distinguishes among the following roles:

- Infrastructure Provider and Infrastructure Installer (IP/II)

  Infrastructure providers and installers are responsible for providing, installing and configuring the hardware elements comprising the smart room. These elements include the full range of sensors, devices, actuators, computing and network equipment required to support sensing, networking and basic services in the smart room. All these hardware elements are – upon their installation – registered with the knowledge base. Registration information includes descriptive elements about vendors, models, interfaces, capabilities, network addresses, as well as the means (*e.g.*, API) to access their capabilities. A possible standardisation of the registration process will allow manufacturers (and infrastructure providers) to implement components that register within the room-wide ontology management system.

- Perceptual Components Provider and Developer (PCP/PCD)

  Perceptual components providers develop, install and configure perceptual components. The latter process sensorial input based on the range of sensors available in the smart space. Assuming that the IP/II has populated the registry with information on sensors, actuators, and other hardware elements, the PCP/PCD can consult this information towards configuring and registering its components.

- Situation Modeller and Situation Model Developer (SM/SMD)

  Situation model developers build higher-level contextual models that rely on recognition of situations (*e.g.*, a peer-to-group human interaction). Situation recognition is likely to be based on a combination of a variety of perceptual interfaces. The later can, for example, be combined according to the network of situations approach (Crowley, 2003). SM/SMDs can therefore rely on descriptions of perceptual components and infrastructure elements to model situations. In particular, information about infrastructure elements and perceptual components defines the context cues that can be provided by the smart space (*e.g.*, the ability to identify persons and their locations, the ability to track persons and artefacts, the ability to access the status of in-home devices, *etc.*). SM/SMDs can then combine these context cues towards building situation models that can identify higher-level contextual states. Situation models are also registered in the knowledge base. Moreover, it includes pointers to the sensors, actuators and perceptual components supporting the situation model. During the run-time parsing of the situation model, these supporting components will be dynamically discovered and invoked based on information provided by the directory service.

- Service Providers/Service Developers (SP/SDs)

  Service providers and service developers deploy and develop (respectively) ubiquitous computing services leveraging situation models, perceptual components, hardware elements, as well as other actuating services. Situation models define the context triggers (*i.e.*, contextual states) that drive the service logic of the application. Perceptive interfaces, actuating services and other service-related actions are accordingly combined to compose the service logic that is executed on each context state.

Note that service actions are also registered with the knowledge base. Service actions are associated with all component types, namely:

- infrastructure elements in order to control, tune or configure sensors, actuators, devices, *etc.*
- perceptual components in order to configure the perceptual component for optimal performance or even to control it (*e.g.*, start/stop it) during the course of a service

- situation models, towards dynamically adapting (*e.g.*, augmenting, restricting or evolving) a situation model
- services, to leverage actuating services (*e.g.*, display a presentation, play a prompt and generally invoke soft-services).

The introduced model supports both the deployment and run-time operation phases of ubiquitous computing services. During the development and deployment phases, software, hardware and middleware components are registered with the knowledge base. This enables developers and deployers to automatically discover the room capabilities in terms of sensors, actuating devices and perceptual components. As already outlined, the latter components serve as a basis for recognising situation and building applications. Hence, SM/SMDs can develop situation models based on the registered capabilities of perceptual components and infrastructure elements. Following the development of the SMD, the situation model is registered in the room directory services. Accordingly, SP/SDs can consult the knowledge base for available situation models, and use them to develop the service logic as well as to deploy the service.

During the run time operation of the system, components may access the knowledge base in order to discover resources and services, as well as to acquire references on them. For example, a situation model or service may look up the knowledge base to locate/discover a tracking component (*e.g.*, a visual tracker or an acoustic person tracker (Pnevmatikakis and Polymenakos, 2005; Stergiou *et al.*, 2005; Talantzis *et al.*, 2005). Accordingly, the situation model can obtain a reference on the tracker towards accessing its capabilities (*i.e.*, its API). Similarly, a service logic components can look up and acquire references to actuating services.

The deployment model depicted in Figure 1 enables interchangeability among perceptual components, situation models, infrastructure and service elements. This is because the knowledge base allows different technology providers, developers and manufacturers to have a common vocabulary for their components. Components from different vendors can be interchangeable as soon as they provide the same functionality, take input and deliver information in the same format.

3 Knowledge base using web ontologies and semantic web technologies

3.1 Knowledge base and ontology management

One may argue that the registration, binding and discovery actions entailed in the introduced deployment model can well be supported through conventional directory services (Balazinska *et al.*, 2002; Czerwinski *et al.*, 1999; Zhu *et al.*, 2003; Guttman *et al.*, 1999),[1–4] Adjie-Winoto *et al.*, 1999). Indeed, technologies such as Universal Plug n' Play (UPnP),[2] Service Location Protocol (SLP) and Universal Description, Discovery and Integration (UDDI)[4] provide mechanisms for registering and discovering resource and services. However, these mechanisms are not particularly tailored to the range of information and components that are essential to Ubicomp services. For example, UDDI and SLP are merely service-oriented, while UPnP is device-oriented. To some extent, this is also due to a lack of standardised descriptions for perceptual components, situation models and Ubicomp services. Therefore, existing directory service mechanisms are not appropriate for supporting the diverse components entailed in ubiquitous computing environments.

Furthermore, the context-aware, human-centric, pervasive nature of ubiquitous computing services, asks for intelligence in answering queries. Intelligence lies in the ability to infer information from existing sets of meta-data according to current context and user intention. As an example, given the number of different sensors in a smart space, a particular situation model or Ubicomp service may need to acquire a reference to the best-view camera for a particular situation, *e.g.*, the camera facing the door for recognising a person entering a room. The ability to answer such queries requires inferring information based on existing meta-data, and on a reasoning procedure. Legacy directory mechanisms rely on a given set of meta-data and do not therefore provide reasoning capabilities. As a result, web ontologies are deemed more appropriate for supporting knowledge management in the introduced deployment model.

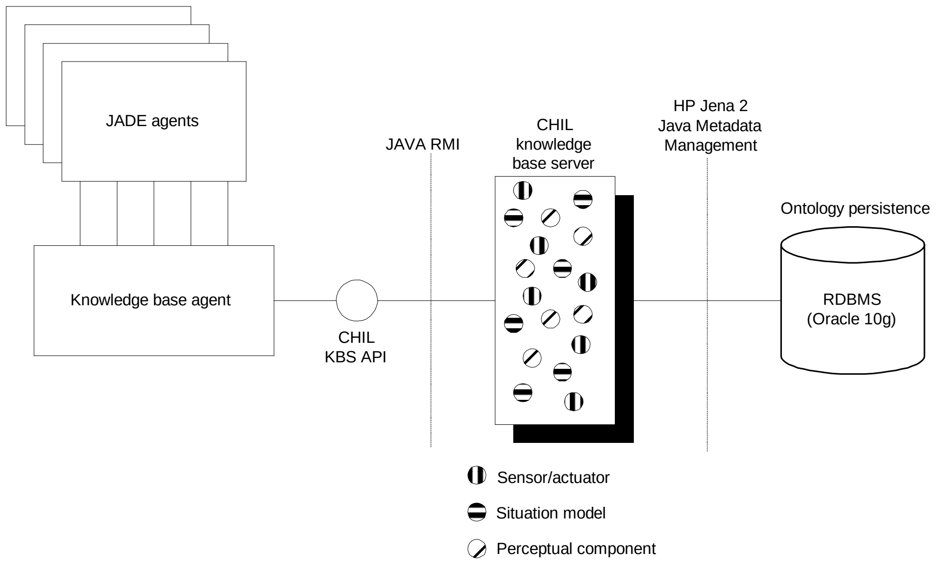
The above argumentation advocates that an ontology management system is the most appropriate choice for implementing the mechanisms for registering, discovering and obtaining references to components. Such an ontology management system can also facilitate communication between distributed entities, which constitutes another benefit over conventional systems. Communication between distributed entities is a mandatory function in the highly distributed and heterogeneous Ubicomp systems. The ontology can boost this process through modelling both the type and value of the content of the messages, which are exchanged among the various distributed systems.

3.2 Knowledge base server

To underpin the value of ontology management systems for the development and deployment of ubiquitous computing application, we have used the CHIL Knowledge Base Server (KBS) (Pandis *et al.*, 2005), which has been developed for the Computers in the Human Interaction Loop (CHIL) project[11] as an adapter for a variety of off-the-shelf ontology management systems for OWL (Web Ontology Language)-based ontologies. Hence, based on the CHIL KBS, Ubicomp service developers and deployers can easily utilise an ontology management technology that best suits their needs. In the scope of the prototype implementations illustrated in the next section, we exploited the CHIL KBS version 1.3.1 along with the JENA2 ontology management system. Note also that the CHIL KBS is implemented as an Eclipse project, which makes it particularly easy to develop both the software application and the underlying ontology in an integrated, convenient and effective way.

Another important characteristic of CHIL KBS implementation is that it supports a variety of remote access protocols, including Java RMI (Remote Method Invocation), SOAP-based web services, as well as other kinds of XML-over-TCP technologies. As a result, client applications can access the server based on a variety of programming languages (*e.g.*, Java, C#). The variety of client access interfaces include also an interface to Java Agent Development Environment (JADE) agents.[12] In interfacing JADE agents with the KBS, a so-called Knowledge Base Agent (KBA) JADE agent has been implemented. The KBA is a wrapper that enables the communication and integration of the server with other JADE agents. KBA instances invoke the Knowledge Base services based on the RMI access interface, as depicted in Figure 2.

Figure 2 Agent (JADE) access interface to the knowledge base server



The JADE interface to the service is used to connect the knowledge base to the CHIL multi-agent middleware infrastructure (Soldatos *et al.*, 2005b), which has been built to facilitate the development of ubiquitous computing services within the CHIL project. Specifically, this multi-agent framework (described in detail in Soldatos *et al.*, 2005a) boosts the reusability of components, and establishes a common methodology for the development of services in a ubiquitous computing environment. The framework provides a set of basic services within each smart room, like mechanisms to control the various sensors and actuators in a common way. Moreover, it controls user access to services, while also facilitating service personalisation by maintaining and managing profiles. Furthermore, it enables the integration of new services, based on a 'pluggable' mechanism. The KBS provides directory services to all agents of this framework, with a view to locating components and services. All the agents in the framework take advantage of the knowledge base ontology management functionality through the JADE access interface and the KBA (Figure 2).

3.3 Knowledge base structure and reasoning

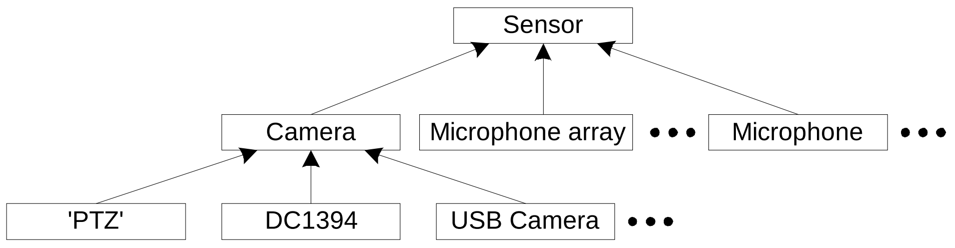
A knowledge base has been built to support the model depicted in Figure 1. The ontology features a modular structure allowing service developers and providers to load and use the parts of their interest/choice. A core part of the ontology introduces the concepts of perceivable entities in the scope of a ubiquitous computing environment (*e.g.*, Person, Meeting Room, Table or Whiteboard), as well as perceivable roles of such entities (*e.g.*, the Location of a Person or the Activity Level of a Room). At the same time, the core module provides a vocabulary for data formats such as audio or video streams.

Having such a core module at hand, another part of the ontology is dedicated to establishing the common vocabulary for perceptual components. This vocabulary describes the information that a perceptual component can deliver (*e.g.*, coordinates of a person), in terms of structure established in the core module. This allows perceptual components to be functionally characterised and looked up by their interface input and output data types, regardless of their level of abstraction and internal structure.

With respect to situation modelling, the ontology adopts a chronological arrangement of events, based on temporal logic. Assuming that precise points in time cannot be modelled anyway, the ontology bases the chronological relation of events on intervals that may touch, overlap, be included, or stay side by side without touching each other. This is currently not possible with plain OWL. However, there are several approaches underway, which aim at adding rules to OWL. Currently, rule-based reasoning itself is done outside of the ontological framework. Still, however, the ontological framework can store ontological data that an external rule-based reasoning system can access. Note that there are algorithms that provide sound and complete reasoning for the description logic (Bader *et al.*, 2003). Reasoning is important to ensure the quality of an ontology and to fully exploit its rich structure. Though we do not rely on ontological reasoning in the scope of our applications, our overall architecture supports a future implementation.

The ontology also includes information on the sensing devices that are available within our smart room. Sensor and actuator information typically includes vendor, model, status of the device, interfaces, capabilities and network addresses. More specific information is added depending on the capabilities of the sensor or device. Sensor information has been hierarchically structured to include information about generic sensors, as well as the more specific microphones, microphone arrays and cameras that are used in CHIL. The hierarchy implemented and included in the ontology is depicted in Figure 3. This hierarchical structure allows for flexibly extending the ontology with additional sensors, since developers need to simply extend existing entities with additional meta-data. Note that the sensing infrastructure within the AIT smart room includes an NIST Mark III: 64 channel microphone array (Brandstein and Ward, 2001), four microphone clusters consisting of four inverted T-shaped SHURE microphone clusters, four fixed Firewire digital cameras, one active camera with pan, tilt and zoom (PTZ camera), as well as a panoramic (or fish-eye) surveillance camera. The ontology is able to instantiate and include information about all these elements.

Figure 3 Typical hierarchy of logical sensor



Apart from information on sensors, the ontology also includes entities capturing the various actuating actions. These entities comprise meta-data relating to actuating services capabilities, input and output parameters, as well as Uniform Resource Identifiers (URI) enabling access to the service. Note that service actions are also available as part of entities relating to perceptual components, situation model and sensors, as already described in previous paragraphs.

3.4 Ontology construction and loading

For each different smart space, a distinct instance of the OWL ontology comprising the knowledge-based needs to be constructed and instantiated. A particular instantiation of the ontology for a smart space is expected to include infrastructure elements and perceptual components. The ontology will therefore be built according to the sensors, devices and perceptual components available within the smart space. Given a number of sensors and perceptual components, situation model developers can build several situation models. Each situation model can capture a different context modelling scenario, which can be supported by the underlying perceptual components and infrastructure elements. The ontology can be incrementally extended with additional situation models provided by the situation modellers and situation model developers. Overall, building the ontology for a given smart space requires the contribution of the infrastructure provider, the perceptual components provider, as well as the situation model developer.

Once the ontology is built, it can be instantiated within the knowledge base. The ontology is therefore loaded upon the start-up of the overall ubiquitous computing system. Accordingly, the ontology can be used for resource discovery and resource management, as outlined in later paragraphs. Note, however, that the ontology is likely to be updated to account for new sensors and perceptual components, as well as for the development and deployment of situation models and services. New sensors and perceptual components may be installed into the smart space to augment its sensing and context-capturing capabilities. Moreover, additional situation models can be used within the smart space to support new scenarios and applications. In general, sensors and perceptual components within a smart space will change at coarser time scales than situation models. No matter what the time scale is, accommodating implies a need for updated and reloading the ontology. In the case of new device types, the augmented ontology will allow new devices to be registered to the ontology. Similarly, tracking new contextual states (*i.e.*, new situation modelling scenarios) requires adding or augmenting situation models within the ontology.

It is noteworthy that there is no standard ontology for context-awareness in smart spaces. At this stage the ontology will be built in a proprietary fashion. A possible standardisation of an ontology for smart spaces could specify a well-defined and unanimously accepted way for describing infrastructure elements, perceptual components and situation models within a smart space. Such a standard (*e.g.*, within W3C or OASIS) would obviate the need for specifying a (RDF/OWL) ontology for use within each different instance of a smart space. Rather it would allow different instantiation of the same ontology structure.

4 Prototype tools and services

We have leveraged the benefits of the KBS and the underlying ontology management system in a number of prototype applications, services and tools relating to ubiquitous computing and smart spaces. Building prototype systems that use the ontology entails the task of registering (hardware, software and middleware) components, continuously updating the ontology with respect to the status of these components, as well as querying the ontology. These applications and tools provide a practical underpinning of the value of the introduced deployment model and the associated ontological knowledge management mechanisms.

4.1 Smart space resource manager

The Smart Space Resource Manager (SSRM) is a utility enabling monitoring and control of perceptual components, sensors and actuating services available in a smart space (*e.g.*, smart room). Administrators of the smart space may use the SSRM utility to manage a variety of heterogeneous and distributed entities from a single entry point. SSRM has been implemented as a multi-agent system, with a twofold objective:

1 to facilitate interoperability between the diverse hardware (*i.e.*, sensing and actuating devices) and middleware (*i.e.*, perceptual and actuating components)

2 to leverage the JADE access interface to the KBS, along with the KBA.

The main building blocks enabling the SSRM are depicted in Figure 4, and include:

- JADE proxies for sensors, actuators and perceptual components

  A JADE proxy is developed for each sensor, actuator or perceptual component to be managed by the SSRM. This agent proxy acts as a gateway allowing other JADE agents (such as the KBA) to interact with the device. Note that proxy agents provide a level of abstraction: each proxy exposes a universal virtualised interface to the agent framework (*i.e.*, a common interface for sensors, actuators, perceptual components of the same type).

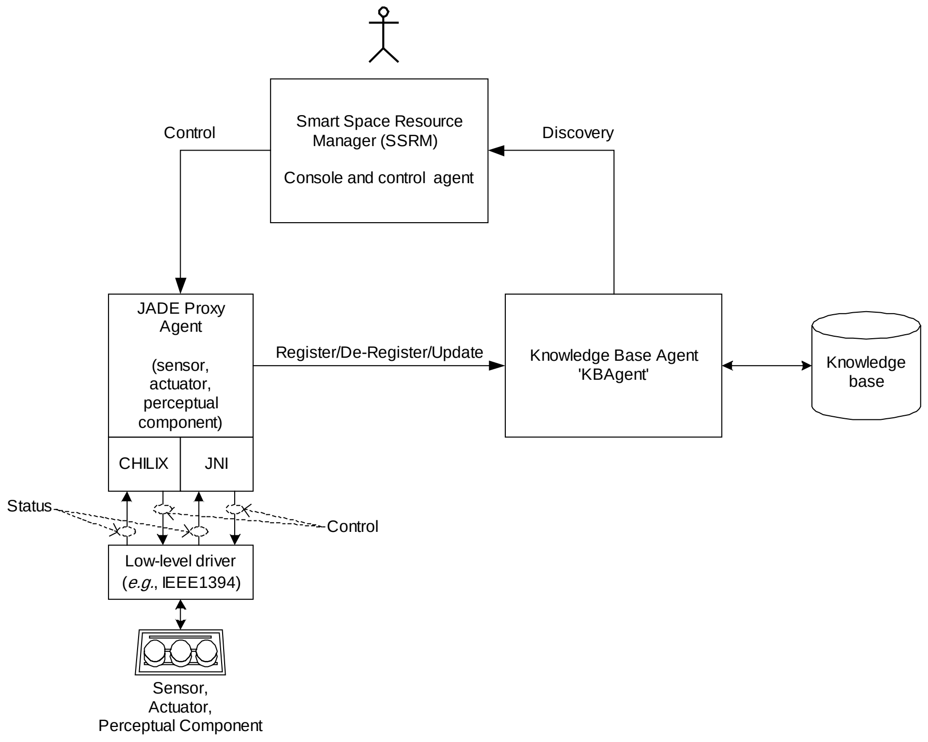
- Low level device drivers

  Every JADE proxy needs to have access to the low-level capabilities of each device or component in order to: (a) obtain information on the device status and (b) issue control commands to the devices. Low-level drivers provide access to the proprietary low-level functionality of the device. The implementation of the low-level driver is device-specific. As an example, access to digital Firewire cameras is based on the IEEE1394 API, while access to perceptual component capabilities depends on the technology provider of the component.

- SSRM control agent and console

  This is the administrative console that enables monitoring and control of the various components. Monitoring functionality is provided by querying the KBS, while control functionality is based on issuing control commands to the device (or component). Control commands change the status of the device and result in a subsequent update of the ontology. Note that the SSRM console is also implemented as a JADE agent to facilitate interoperability and communication with the KBA and JADE proxies.

Figure 4 Building blocks of the Smart Space Resource Manager (SSRM)


Smart Space Resource Manager (SSRM)
Console and control agent
Control
Discovery
JADE Proxy Agent
(sensor, actuator, perceptual component)
Register/De-Register/Update
Knowledge Base Agent 'KBAgent'
Knowledge base
CHILIX
JNI
Status
Control
Low-level driver (*e.g.*, IEEE1394)
Sensor, Actuator, Perceptual Component

- KBA JADE agent

  The KBS agent provides the required interfacing to the KBS. It responds to register, de-register and update events, which are issued by proxy agents.

Note that the interface between a proxy agent and the device or component requires interfacing Java with native (*e.g.*, C/C++) code. In the scope of our implementation, we have relied in the Java Native Interface (JNI), as well as the CHILIX module, an IBM library providing high-performance XML-over-TCP access to perceptual components and other low-level drivers.

The current implementation of the SSRM provides monitoring functionality for a variety of sensors, actuators and perceptual components. It can graphically depict the content of KBS ontology through an easy-to-use interface. In particular, the SSRM dynamically retrieves the registered elements and presents them in a dynamic GUI tree. Moreover, it can also issue control commands to the proxy agent of FireWire cameras. Control commands (*e.g.*, changing capture frame-rates, changing zoom levels) on these cameras are handy in cases where there is a need to adapt the camera operation to either a particular context or to the operational requirements of a perceptual processing component (*e.g.*, a face recognition or visual tracking algorithm).

The SSRM is a useful tool that can support several processes entailed in the development and deployment cycle of ubiquitous computing applications. Monitoring of sensors, actuating devices and perceptual components is important for developing and debugging ubiquitous computing services, since it facilitates identification of component failures and failing factors. Moreover, it can facilitate deployment by providing a bird's eye view of the infrastructure and the perceptive capabilities of a smart space.

4.2 Infrastructure ambient room services

Based on the registration of actuating services in the ontology, we have also implemented an infrastructure for ambient services. This infrastructure allows Ubicomp applications to incorporate services that automatically select the best flavour of a particular actuating service. Characteristic examples are:

- An intelligent cameraman service, automatically selecting the camera that provides the optimal view of a speaker (as described in Azodolmolky *et al.*, 2005).
- An intelligent display service, automatically selecting the display that is most appropriate for a target person within the smart space.

The infrastructure for such ambient services relies on the part of the ontology that includes high-level descriptions of the actuating services (*e.g.*, text-to-speech (TTS) services, display services) that are available in the smart space. Based on these descriptions, a mechanism for controlling these smart room services has been implemented. The implementation is similar to the proxy-based control of the sensors, devices and components implemented as part of the SSRM. Specifically, (JADE) proxy agents have been implemented to represent actuating services in the agent world (*i.e.*, also enabling communication with the rest of the framework). Similar to the SSRM case, proxy agents for actuating services translate requests from the various clients to service-specific calls and interact with the KBS (through the KBA). All services that provide the same functionality (*e.g.*, all the TTS services) are wrapped by a service proxy of this type (*e.g.*, a TTS proxy). Hence, all the requests for such a service are forwarded to the proxy of that service, which determines the specific implementation that will handle the request. The proxy retrieves (from the knowledge base) run-time information about the properties and the operational status of the actual services and then takes the decision.

Note that the implemented infrastructure provides capabilities just for locating and invoking services. The algorithm for deciding which service is the most appropriate for a request needs to be incorporated in the proxy implementation. For example, in the case of several different operational TTS service instances, the TTS service proxy would decide to forward the request to the TTS device that is 'nearer' to the service target; while a display or targeted-audio service proxy would decide to forward the request to the machine 'opposite' the target. Implementing this decision logic is greatly facilitated by ontological reasoning. Indeed, ontological reasoning can flexibly provide the devices that are *nearer* to or *opposite* a target, given a set of topological meta-data and rules about the smart room and its devices. Answering such queries using the KBS is rather straightforward, which is a clear advantage of the ontology mechanism over conventional directory services. The latter would require much more development effort for answering the same queries.

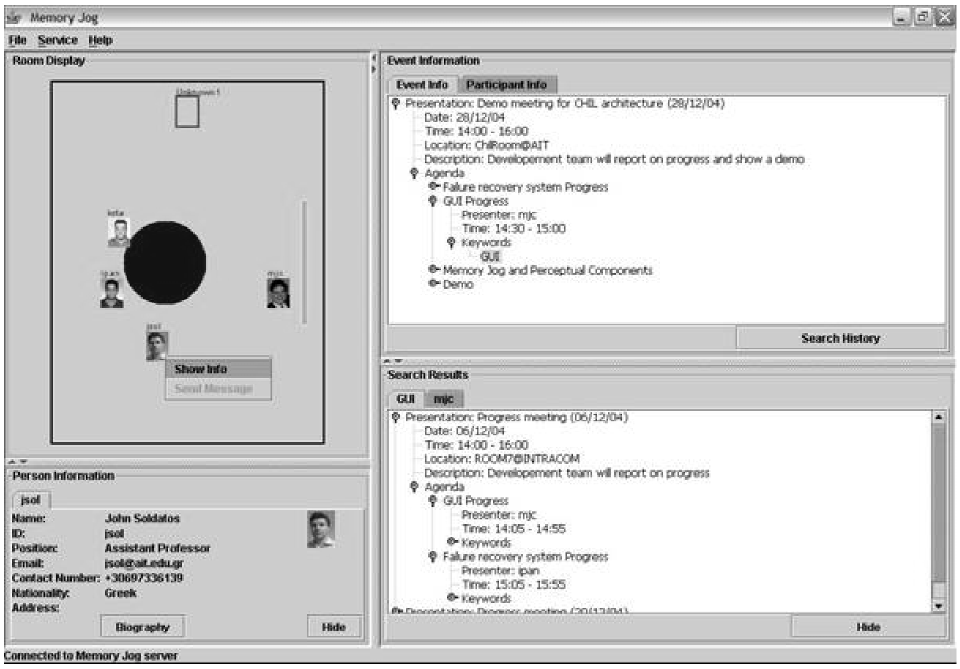
4.3 The memory jog: prototype ubiquitous computing service

The memory jog is a prototype ubiquitous computing service aiming at supporting meetings, lectures, presentations and conferences in smart spaces (Soldatos *et al.*, 2005b). In the scope of this service we have implemented functionality for tracking human activities within a room and accordingly providing information to meeting participants. Information is provided continually both when users request it, but also automatically based on context. Information pertains to people information (*e.g.*, biography), as well as their activities in a meeting, such as the topics, keywords and agenda items. The service can capture situations such as the start and the end of a meeting (or lecture), the start of a presentation, questions during a presentation, as well as changes in agenda items. Situations are linked together based on the network of situations approach (Soldatos *et al.*, 2005a). A snapshot of the services graphical user interface is depicted in Figure 5, along with the network of situations that can be identified and tracked by the service.

The memory jog service relies on all the sensors of our smart room (Soldatos *et al.*, 2005a), as well as on a range of perceptual components. Specifically, an audio source localisation system (Talantzis *et al.*, 2005) provides speaker location and tracking. Participant locations within the smart space are provided based on our visual person tracking technology. Moreover, a face detection, eye tracking and face recognition system (Stergiou *et al.*, 2005; Talantzis *et al.*, 2005) allows for identification of the meeting participants. A speech activity detection system allows the system to infer when participants engage in speech activity. Finally, an agenda-tracking component signifies advancement in the agenda. Such advancements rely on the particular scenario, which assumes that the agenda advances whenever a new speaker stands close. Perceptual components, sensors and actuating devices entailed in this service have been integrated based on the above-mentioned multi-agent middleware infrastructure for ubiquitous computing applications. Note that a thorough description of the technical architecture of the memory jog service can be found in Soldatos *et al.* (2005a).

Early instantiations of this architecture did not make any use of the knowledge base, since the CHIL KBS was under development. Thus, integration between perceptual components, sensors and situation models relied on hard-coded communication between the respective middleware components. Since the introduction of the CHIL KBS in the AIT, Smart Room integration has become much easier and straightforward allowing developers to focus on improving the capabilities of the underlying perceptual components, as well as on evolving the service logic of the memory jog. The use of the KBS service allowed the memory jog service to dynamically discover and invoke audio- and vision-based person tracking components, which accelerate development, facilitate integration and overall guarantee technological longevity.

Figure 5 Graphical user interface and situation model for the memory jog service

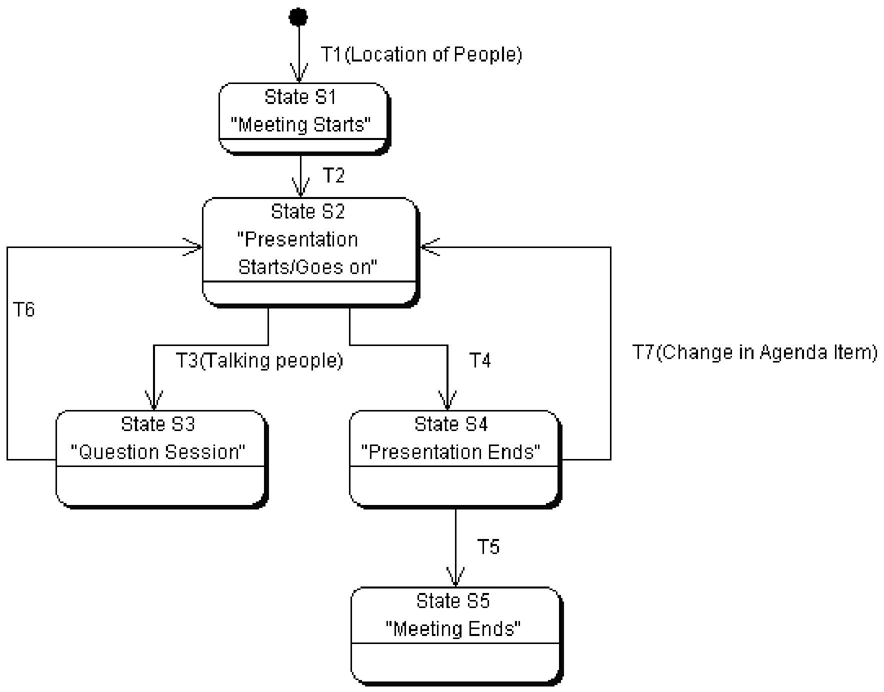


```
Presentation: Demo meeting for CHIL architecture (28/12/04)
    Date: 28/12/04
    Time: 14:00 - 16:00
    Location: ChilRoom@AIT
    Description: Developement team will report on progress and show a demo
    Agenda
        Failure recovery system Progress
        GUI Progress
            Presenter: mjc
            Time: 14:30 - 15:00
            Keywords
                GUI
        Memory Jog and Perceptual Components
        Demo
```

```
Presentation: Progress meeting (06/12/04)
    Date: 06/12/04
    Time: 14:00 - 16:00
    Location: ROOM7@INTRACOM
    Description: Developement team will report on progress
    Agenda
        GUI Progress
            Presenter: mjc
            Time: 14:05 - 14:55
            Keywords
        Failure recovery system Progress
            Presenter: ipan
            Time: 15:05 - 15:55
            Keywords
Presentation: Progress meeting (20/12/04)
```

Name:	John Soldatos
ID:	jsol
Position:	Assistant Professor
Email:	jsol@ait.edu.gr
Contact Number:	+30697336139
Nationality:	Greek
Address:	

Connected to Memory Jog server



The potential benefits of using the memory jog in real meetings, conferences and presentation include faster task completion, less time and better accuracy to access past information, less time to produce and approve minutes, as well as easier scheduling of future meetings and presentations. We have observed several of these benefits based on focus groups that have been organised to evaluate the service. These groups have proven that users enjoyed more productivity and satisfaction over conventional meetings, lecture and presentations. The memory jog service belongs to the broad class of non-obtrusive context-aware services that are gradually moving from the laboratories to the enterprise (Brown *et al.*, 1997). Such services manifest a new paradigm where services assist humans in a transparent non-intrusive fashion, minimising the need for humans to operate in the computer's loop. Note, however, that such services are still developed and used in the research labs, rather than in realistic applications. This is because a host of challenges need yet to be resolved towards their deployment. These challenges relate to technical issues (*e.g.*, robustness of perceptual components, multi-modal interfaces and recognition algorithms), as well as to social and people issues (*e.g.*, how humans accept/react to this novel non-obtrusive paradigm).

5 Conclusion

Ubiquitous computing services comprise rich collections of hardware, software and middleware components. Furthermore, they are characterised by extreme diversity and dynamism, which make service development a challenging task. Naming and directory services are a key prerequisite to accelerating and facilitating development and deployment of ubiquitous computing services. While a large number of legacy naming and directory mechanisms exist, we strongly believe that they do not successfully address the challenges of a ubiquitous computing environment, since their scope cannot cover the full range of components that are required to support sophisticated ubiquitous computing services. Having this limitation, as a starting point we propose alternative naming and directory mechanisms based on Web Ontologies (OWL). Ontologies provide a much wider scope in conceptualising knowledge. Moreover, they provide reasoning capabilities, which enable intelligent query answering mechanisms. Intelligent query answering is not provided in the scope of conventional naming and directory technologies, yet it is extremely valuable to context-aware pervasive environments.

This paper has introduced a development and deployment model for ubiquitous computing applications. This model distinguishes the roles of entities that develop and deploy infrastructure elements, perceptual components, situation models, as well as service logic. We have outlined how this model can be supported by a web ontology. As proof of concept, we have set up an ontology management system, which manipulates an ontology that is in line with the presented deployment model. Specifically, this ontology includes modules and entities that conceptualise knowledge on perceptual components, sensors, actuating services and situation models. The benefits of this ontology management system for developing and deploying Ubicomp applications are manifested in three different real-life applications that operate in a realistic smart space (*i.e.*, smart room):

1 A Smart Space Resource Manager, facilitating monitoring and control of perceptual components, sensors and actuating devices. This tool constitutes a proof-of-concept of the web ontology as a generalised directory service for ubiquitous computing.

2 An infrastructure for implementing ambient services, which demonstrates the power of reasoning on the ontology.

3 A prototype ubiquitous computing application, which uses the ontology management system for flexible dynamic (*i.e.*, run-time) access to resources and components within the smart space.

All these applications have been implemented based on the capabilities of an agent platform, namely, the JADE platform. Agents facilitate the development of ubiquitous computing applications by supporting transparent distributed communications, context-modelling and acquisition, as well as a fault-tolerance and other autonomic features (Tesauro *et al.*, 2004). Moreover, agents allow for a fine integration with ontologies and ontology management systems, for both reasoning and inter-agent communications (Soldatos, 2006). Therefore, even though alternative implementation technologies exist, agents seem to offer significant benefits.

The contributions of this paper can be seen as a first step in the direction towards the standardisation of meta-data relating to sensors, perceptual components, actuating devices and services. Such a potential standardisation could essentially boost the deployment of ubiquitous computing applications, which are currently very hard to build and are barely portable across different smart spaces and infrastructures. The proposed approach is in line with the overall vision of intelligent context-aware plug ‘n play services (Neuhold *et al.*, 2005; Fuchs *et al.*, 2003), which exploit emerging knowledge inference and reasoning capabilities (*e.g.*, as in Ceri *et al.*, 2003).

Acknowledgements

This work is part of the FP6 CHIL project (FP6-506909), partially funded by the European Commission under the Information Society Technologies (IST) programme. The authors acknowledge the valuable help and contributions from all partners of the project. Special thanks to Alexander Paar and Jürgen Reuter (Institute for Program Structures and Data Organisation, University of Karlsruhe), who implemented the knowledge base server, as well as a variety of access methods to it.

References

Adjie-Winoto, W., Schwartz, E., Balakrishnan, H. and Lilley, J. (1999) ‘The design and implementation of an intentional naming system’, *Symposium on Operating Systems Principles*, pp.186–201.

Azodolmolky, S., Dimakis, N., Mylonakis, V., Souretis, G., Soldatos, J., Pnevmatikakis, A. and Polymenakos, L. (2005) ‘Middleware for in-door ambient intelligence: the PolyOmaton system’, *Proc. of the 2nd Next Generation Networking Middleware Workshop (NGNM ’05)*, held in the scope of *Networking 2005 Conference*, Waterloo, Canada, May.

Bader, F., Calvanese, D., McGuiness, D., Nardi, D. and Patel-Schneider, P. (2003) *The Description Logic Handbook*, Cambridge University Press.

Balazinska, M., Balakrishnan, H. and Karger, D. (2002) 'INS/Twine: a scalable peer-to-peer architecture for intentional resource discovery', *Proceedings of the First International Conference on Pervasive Computing*, Springer-Verlag, pp.195–210.

Brandstein, M. and Ward, D. (Eds.) (2001) *Microphone Arrays: Techniques and Applications*, New York: Springer-Verlag.

Brown, P.J., Bovey, J.D. and Chen, X. (1997) 'Context-aware applications: from the laboratory to the marketplace', *IEEE Personal Communications*, October, Vol. 4, No. 5, pp.58–64.

Ceri, S., Gennaro, C., Paraboschi, S. and Serazzi, G. (2003) 'Effective scheduling of detached rules in active databases', *IEEE Transactions in Knowledge Data Engineering*, Vol. 15, No. 1, pp.2–13.

Crowley, J.L. (2003) 'Context driven observation of human activity', *Proc. of the European Symposium on Ambient Intelligence*, October.

Czerwinski, S.E., Zhao, B.Y., Hodes, T.D., Joseph, A.D. and Katz, R.H. (1999) 'An architecture for a secure service discovery service', *Mobile Computing and Networking*, pp.24–35.

Dey, A.K., Salber, D. and Abowd, G.D. (2001) 'A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications', *Human-Computer Interaction*, Vol. 16.

Fuchs, M., Niederée, C., Hemmje, M. and Neuhold, E.J. (2003) 'Supporting model-based construction of semantic-enabled web applications', *Prof. of WISE*, pp.232–241.

Guttman, E., Perkins, C., Veizades, J. and Day, M. (1999) 'RFC 2608: service location protocol, version 2', http://www.ietf.org/rfc/.

Johanson, B., Fox, A. and Winograd, T. (2002) 'The interactive workspaces project: experiences with ubiquitous computing rooms', *IEEE Pervasive Computing Magazine*, April–June, Vol. 1, No. 2.

Möller, R. and Haarslev, V. (2006) *RACER: Renamed ABox and Concept Expression Reasoner*, Hamburg, http://www.sts.tu-harburg.de/~r.f.moeller/racer/.

Neuhold, E.J., Risse, T., Wombacher, A., Niederée, C. and Mahleko, B. (2005) 'Intelligent web service – from web services to .Plug&Play. service integration', *Proc. of the OTM Conferences*, Vol. 1, pp.18–19.

Pandis, I., Soldatos, J., Paar, A., Reuter, J., Carras, M. and Polymenakos, L. (2005) 'An ontology-based framework for dynamic resource management in ubiquitous computing environments', *2nd International Conference on Embedded Software and Systems*, Northwestern Polytechnical University of Xi'an, P.R. China, 16–18 December.

Pnevmatikakis, A. and Polymenakos, L. (2005) 'An automatic face detector and recognition system for video streams', *2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, Edinburgh, July.

Smailagic, A. and Siewiorek, D.P. (2002) 'Application design for wearable and context-aware computers', *IEEE Pervasive Computing*, December, Vol. 1, No. 4, pp.20–29.

Soldatos, J. (2006) 'Software agents in ubiquitous computing: benefits and the CHIL case study', *Proc. of the Software Agents in Information Systems and Industrial Applications (SAISIA'06) Workshop*, Karlruhe, Germany, 9–10 February.

Soldatos, J., Pandis, I., Stamatis, K., Polymenakos, L. and Crowley, J. (2005a) 'A middleware infrastructure for autonomous context-aware computing services', *Computer Communications Magazine, Special Issue on Emerging Middleware for Next Generation Networks*.

Soldatos, J., Polymenakos, L., Pnevmatikakis, A., Talantzis, F., Stamatis, K. and Carras, M. (2005b) 'Perceptual interfaces and distributed agents supporting ubiquitous computing services', *Proc. of the Eurescom Summit 2005*, April, pp.43–50.

Stergiou, A., Pnevmatikakis, A. and Polymenakos, L. (2005) 'Audio/visual person identification', *2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, Edinburgh, July.

Talantzis, F., Constantinides, A.G. and Polymenakos, L.C. (2005) 'Estimation of direction of arrival using information theory', *IEEE Signal Processing Letters*, Spring.

Tesauro, G., Chess, D.M., Walsh, W.E., Das, R., Segal, A., Whalley, I., Kephart, J.O. and White, S.R. (2004) 'A multi-agent systems approach to autonomic computing', *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, New York City, New York, USA, 19–23 July, Vol. 1, pp. 464–471.

Want, R., Hopper, A., Falcao, V. and Gibbons, J. (1992) 'The active badge location system', *ACM Transactions on Information Systems*, January, Vol. 10, No. 1, pp.91–102.

Weiser M. (1991) 'The computer for the 21st century', *Scientific American*, Vol. 265, No. 3, pp. 66–75.

Yau, S.S., Karim, F., Wang, Y., Wang, B. and Gupta, S.K.S. (2002) 'Reconfigurable context-sensitive middleware for pervasive computing', *IEEE Pervasive Computing*, joint special issue with *IEEE Personal Communications on Context-Aware Pervasive Computing*, Los Alamitos, USA: IEEE Computer Society Press, July–September, Vol. 1, No. 3, pp.33–40.

Zhu, F., Mutka, M. and Ni, L. (2003) 'Splendor: a secure, private, and location-aware service discovery protocol supporting mobile services', *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom03)*, March, pp.235–242.

Notes

1 Bluetooth SIG (2001) *Specification of the Bluetooth System, Vol. 1, Core, Rev. 1.1*, Bluetooth SIG, Tech. Rep.

2 UPnP Forum (2003) *UPnP Device Architecture 1.0*, May, http://www.upnp.org/.

3 Sun Microsystems (1999) *Jini Technology Architectural Overview*, Sun Microsystems, Inc., Tech. Rep., http://www.sun.com.

4 UDDI Consortium (2002) *UDDI Technical White Paper*, http://www.uddi.org/pubs/.

5 W3C Recommendation: RDF Primer (2004) http://www.w3.org/TR/rdf-primer/.

6 DARPA's Information Exploitation Office (2001) *DAML+OIL*, http://www.daml.org/2001/03/daml+oil-index.html.

7 W3C Recommendation (2004) *OWL Web Ontology Language Overview*, http://www.w3.org/TR/owl-features/.

8 Stanford University School of Medicine (2003) Protégé knowledge acquisition system, http://protege.stanford.edu/.

9 IBM Alphaworks (2003) *IBM Ontology Management System*, http://alphaworks.ibm.com.

10 HP Labs (2004) *Jena 2 – A Semantic Web Framework*, http://www.hpl.hp.com/.

11 *The CHIL Project*, http://chil.server.de.

12 *Java Agent Development Environment*, http://jade.tilab.com/.