Ontology-Based Context-Aware Service Discovery for Pervasive Environments

Pravin Pawar
Architecture and Services of Network Applications group
Department of EEMCS
University of Twente, The Netherlands
P.Pawar@utwente.nl

Andrew Tokmakoff
Telematica Instituut
The Netherlands
Andrew.Tokmakoff@telin.nl

Abstract— **Existing service discovery protocols use a service matching process in order to offer services of interest to the clients. Potentially, the context information of the services and client can be used to improve the quality of service matching. To make use of context information in service matching, service discovery needs to address certain challenges. Firstly, it is required that the context information should have unambiguous representation. Secondly, the mobile devices should be able to disseminate context information seamlessly in the fixed network. And thirdly, dynamic nature of the context information should be taken into account. The proposed Context Aware Service Discovery (CASD) architecture deals with these challenges by means of an ontological representation and processing of context information, a concept of nomadic mobile context source and a mechanism of persistent service discovery respectively. This paper discusses proposed CASD architecture, its implementation and suggests further enhancements.**

I. INTRODUCTION

Service Oriented Architecture (SOA) is a software architecture approach which assumes a collection of services that communicate with each other to achieve a common goal. The SOA paradigm includes advertising, discovery and utilization of diverse services by means of service directories. The principal components of a SOA consist of a service, service description, service advertising and discovery and artifacts [13]. A service is a contractually defined behavior that can be implemented and provided by a component for use by another component. The service description consists of the technical parameters, constraints and policies that define the terms to invoke the service. The service advertises its service description for potential clients. A client interested in accessing the service obtains information about the existence of a service, its applicable parameters and terms through service discovery. An artifact specifies the associated data model for the service (such as XML schemas and web-service descriptions) to which a client should bind for using the service. A service provider may make an entry into the service directory to reference the artifact and explain how to bind to it. The clients may retrieve this information and use it to bind to the artifacts [13].

Ongoing miniaturization and power efficiency of mobile devices have led to widespread availability of devices that have an increasing amount of processing power and storage, and that support multiple wireless network interfaces connecting to various auxiliary devices and to the Internet. These advances enable mobile devices to consume services published in the service discovery network. However, the existing service discovery protocols match the services considering only the keywords from the user's query and the terms in the service descriptions [2]. These protocols do not consider the context information of the services and clients. Context is a situation of an entity (person, place or object) that is relevant to the interaction between a user and an application [6]. Context-aware computing is a paradigm closely related to mobile computing [5]. Mobile clients usually prefer using services based on several context parameters such as location, time, user identity and profile, device capabilities etc [16]. This indicates that the client and services context information influences the quality of service matching. However, context information is highly interrelated and has many alternative representations that makes it difficult to interpret and use. One possible solution is to use ontologies to specify the interrelations among context entities and ensure common, unambiguous representation of these entities [2].

This work considers that a *context source* provides access to context information [17]. To facilitate scalable development of a context source, storage and retrieval of context information and easy addition of the context source, the context source should provide standardised support for applications. Since a service as part of SOA paradigm provides a well-defined functional behavior, modeling a context source as service should provide a standardised functionality for the development of context-aware applications.

Usually, entities such as service directories are hosted in the fixed network. In contrast, context information such as location, user profile etc. is generally collected from the mobile device used by the client. To facilitate publishing mobile context sources in the fixed network, the necessary framework offering this functionality should address the following challenges: (i) uncertain lifetime of mobile devices (e.g., loss of battery power, loss of connection), (ii) frequent change of used network infrastructures (e.g., switching between ad-hoc and managed networks) and (iii) the role of mobile devices shifts from lightweight clients to data providers.

In the vision of Ambient Intelligence [1], it is expected that homes become places populated by numerous heterogeneous devices that are connected both to themselves and also to the outside world. These devices can be embedded but may also be mobile within the home making use of wireless commnications

technologies. In such a situation, it is reasonable to foresee that homes will make use of a *gateway* device that can both interact with in-home services and also enable interaction between services located in different homes. One candidate technology that can be utilised to realise such a gateway is the OSGi service platform [14].

Based on this motivation, the work reported in this paper focuses on the following topics:

i) Design of a *Context-Aware Service Discovery* (CASD) service which can determine the most suitable service by taking into account the context information of both, the service and client
ii) In the event of a context change or the appearance of new services, if a more suitable service is found, then the client is notified of this more appropriate service by means of a *persistent service discovery* mechanism
iii) Modeling context sources as *services* and providing standardized functionality for updating and retrieving context information
iv) A context source on a mobile device that can participate in the fixed network as a *nomadic mobile context source*
v) Use of *ontologies* for context representation and processing during the service matching process

Section II of the paper describes the CASD model. It briefly explains the role of elements involved in CASD. Section III elaborates on the CASD algorithm and presents an example which further helps to understand the concepts of CASD. Section IV covers technical realization of CASD whilst Section V summarises and proposes future work.

II. Context Aware Service Discovery Architecture

The conceptual model for CASD is shown in Figure 1. In this model, every *service* and *client* may have one or more context sources. The *context source* is a service that provides context information of the associated service or client. The services and context sources register with the *service directory* so they can be discovered. A service has knowledge about its context sources and the *service description* contains a *reference to its context sources*. The *CASD service* is also a service and is discoverable by the client. The client requests for a suitable service with the CASD service. The CASD service retrieves the services matching the service type specified by the client after querying the service directory. Such services are referred as *basic matching services*. *Basic* matched services are those which are returned by a regular discovery service (i.e. protocols that do not consider context information when performing service discovery such as, for example, that proposed in [3]). The CASD service *collects* the context information of the basic services and client. It further *filters* the basic services based on the context information to return *the most suitable service* to the client. Our approach is similar to that outlined in [12], except that we evaluate client context information using the client's context source instead of relying on static *Service Registry* context information.

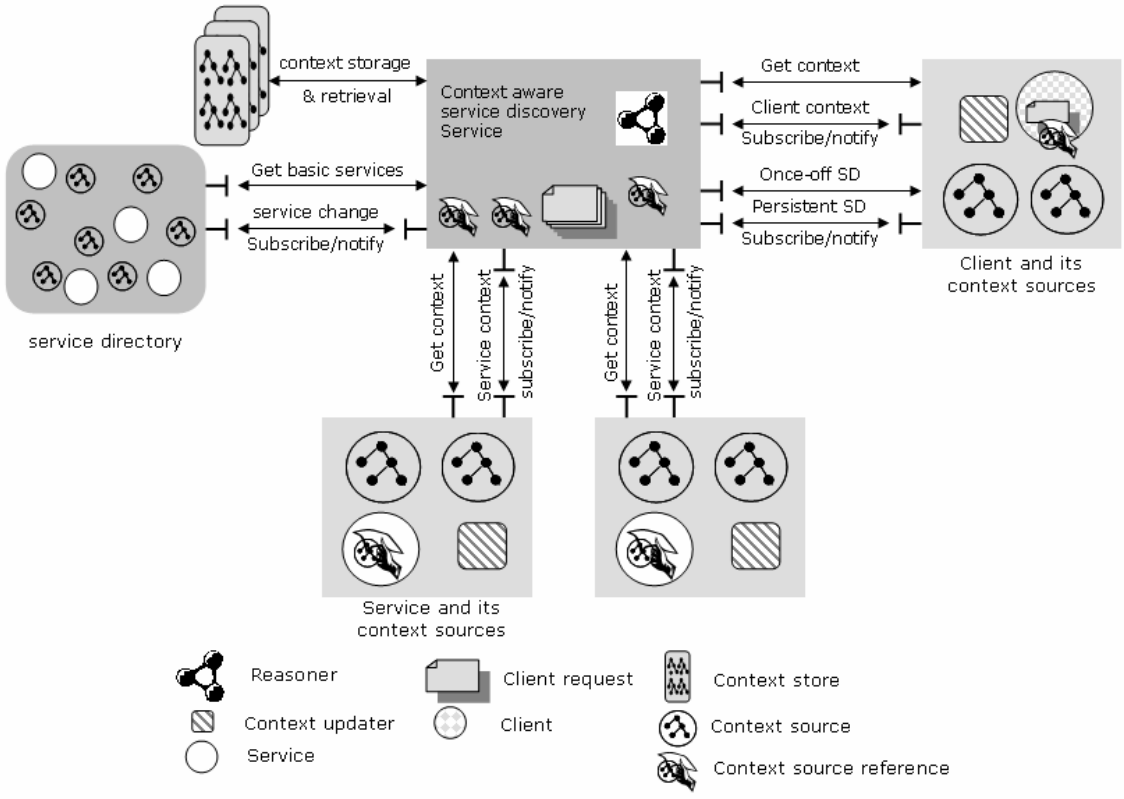

Figure 1. Context Aware Service Discovery Model

The proposed CASD service uses *ontologies* for modeling context information. *Ontology* is an explicit formal specification of how to represent the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them [11]. In our work, ontologies provide a shared vocabulary for specification of client and service context information. Querying the set of services is tightly coupled with the ontologies referred to represent the client and services context information. The following sections explain the elements of the CASD model shown in Figure 1.

A. Context Aware Service Discovery Service

The CASD service provides the following interfaces:

i) *Once-off service discovery*: This interface allows clients to query services based on their context information. The client sends *the type of desired service*, *reference to the client context sources* and *query based on ontological representation of service context* and *client context*. The CASD service returns the service which satisfies the client query.

ii) *Persistent service discovery*: Persistent service discovery is particularly useful when a client is in a dynamic environment and has a need to continually obtain a handle to the *best* service. In this case, in addition to the parameters used with one-time service discovery, the client also registers a *callback interface* with the CASD service. The CASD service *store and processes* the client request whilst the client remains subscribed for persistent service discovery. The CASD service *re-evaluates* the current service sent to the client when triggered by a *client context change*, a *service context change* or when a new service matching the type of desired *service registers* with the service directory. This approach is similar to *reactive discovery* proposed in [4], except in our case, it is the CASD which triggers service discovery re-evaluation based on context changes, rather than the client itself.

B. Context sources

As previously noted, context sources are modeled as services and register with the service directory. A context source provides the following interfaces for its clients:

i) *Get context:* This interface allows clients to obtain the context information. The client should specify the context parameter it is interested in and the context source provides the value of requested parameter.

ii) *Subscribe context*: This interface allows the client to subscribe for the context change event. The client should send a callback interface over which the context change notification is sent.

The context source also provides a *context warehouse* where context information can be stored. *Context updater* is the component which updates the context information at the context warehouse. The context source later distributes this context information to the subscribed clients. The context source, on registration with a service directory provides a *reference to its owner* i.e. service or CASD client. The service and client can further provide this reference to the CASD service for collecting and subscribing context information.

A context source may also be mobile. In the proposed work, a mobile context source is *nomadic mobile service*. A nomadic mobile service (described in [9]) provides the flexibility of allowing a mobile device to participate in the service discovery network and offer these services to the clients located anywhere in the Internet. *Mobile Service Platform* (MSP) proposed in [9] acts as a supporting infrastructure to extend the SOA paradigm to the mobile device. MSP is a middleware that facilitates the development and deployment of nomadic mobile services. MSP addresses the challenges involved in publishing a mobile context source in the fixed network (Pl. refer Section I). A nomadic mobile service

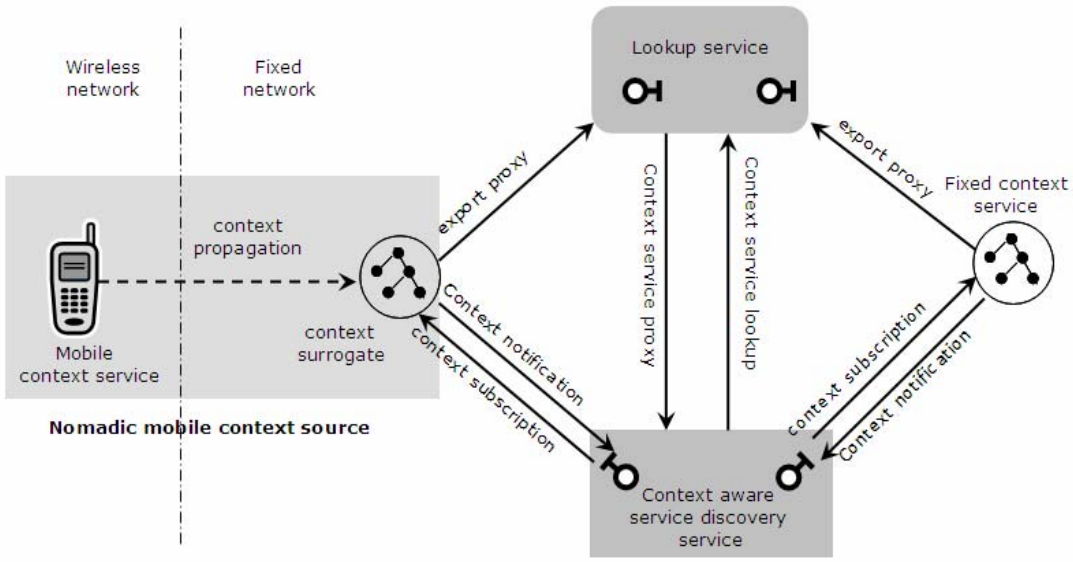

Figure 2. Mobile and Fixed context sources

providing context information is herewith referred to as *nomadic mobile context source*. As shown in Figure 2, a nomadic mobile context source prototyped using MSP is composed of two components: 1) context service running on the mobile device; and 2) representation of the device service in the fixed network which is referred to as a *surrogate*. The other entities shown in Figure 2 will be discussed in Section IV on implementation of CASD.

C. Client

The client of CASD service also has associated context sources. The client is *aware of* the context parameters of a service. Based on this knowledge, the client specifies a query to the CASD service. The query sent by the client describes the constraints that should to be applied to the prospective services (e.g., printing services that should be the closest, and/or the fastest).

D. Reasoner

The *reasoner* component is responsible for *matching* the suitable services based on client context and service context. For this purpose, CASD service sends the context information of all the basic matching services and client to the reasoner. The reasoner retrieves the services which match the client requirements and informs CASD service of its selection.

III. Context Aware Service Discovery Algorithm and Example

This section explains the CASD algorithm and describes an example to explain various concepts referred in this paper. Though this example is simple, it helps to understand the working of proposed CASD algorithm, the use of ontologies and further, it can provide the guidelines for developing complex applications.

A. CASD service algorithm

The sequence diagram for the proposed CASD service is shown in Figure 3. The description of various labeled steps is as follows:

1. The client submits a request to the CASD, indicating the type of service it is interested in, a constraint expression on the matching and also a reference to its context source.
2. The CASD service uses the service directory to obtain a set of basic services that match the requested service type. In case of the request for persistent service discovery, the CASD service registers with the service directory to be notified when a new basic service appears in the network.
3. The CASD service retrieves the service information of each basic service to obtain reference to its context sources.
4. For each of these context sources, the CASD service requests the current context. Additionally, in case of the request for persistent service discovery, the CASD service subscribes to the context change event of every context source.
5. The collected context information of the basic services is stored in a structure called a *service graph*.
6. The CASD service further collects the context information of the client and in case of persistent service discovery, subscribes to the context change event of the client context sources.

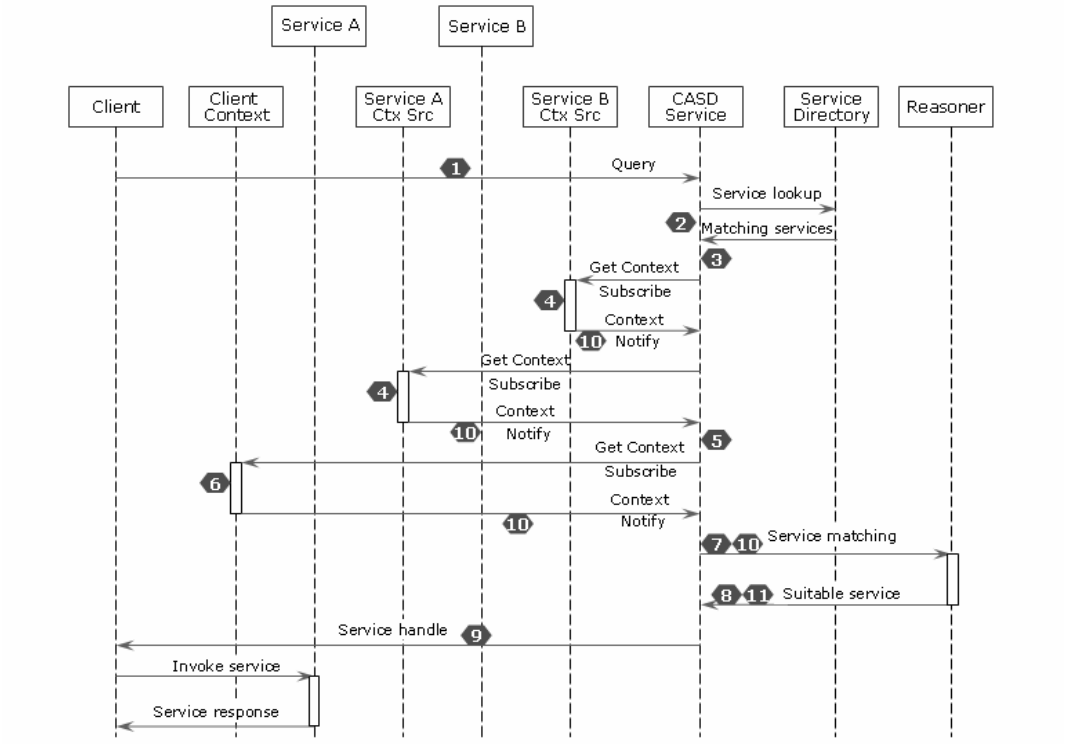

Figure 3. Sequence diagram for context aware service discovery

7. The CASD service populates the client query with client context information and passes query and services graph to the reasoner. In case of persistent service discovery, the service graph is persisted in the database for later retrieval.
8. The reasoner matches the client query with each of the service in the service graph and returns a matching service (if any).
9. The CASD service passes a service reference to the client which can use the service as per service semantics.
10. After receiving a context change event (or when a new basic service appears in the network), the CASD service retrieves the stored service graph from the database and updates the existing service's context information (or appends the context information of new service). The CASD service sends this graph for matchmaking to the reasoner.
11. The reasoner further informs the CASD service if a new service matching client criteria is found.

B. CASD example

The example scenario described herewith is part of a broader scenario chartered by ISTAG [8]. This scenario is outlined as follows:

"Dimitrios, a 32 year-old employee of a major food-multinational, is taking a coffee at his office's cafeteria. Dimitrios is wearing, embedded in his clothes (or in his own body), a voice activated 'gateway' or digital avatar of himself, familiarly known as 'D-Me' or 'Digital Me'. During the coffee time, D-Me 'rings' Dimitro about a call from his wife using a pre-arranged call tone. Dimitrios takes up the call with one of the available Display phones of the cafeteria. D-Me can always point at the closest functioning one display phone!"

This scenario is interesting from the CASD perspective, as *D-Me* can use the CASD to obtain a handle on the closest functioning display phone service. The following components contribute to this CASD scenario:

i) *Display phone Service:* All the display phones in Dimitros' office offer a display phone service. These services register with the service directory so that CASD service can discover them. Display phone offers a service of connecting to other display phone and facilitate a video call between the caller and callee.
ii) *D-Me:* D-Me is a client of the CASD service and uses it to find the closest available display phone (so that Dimitrios can communicate with his wife). An advanced device such as D-Me could potentially provide various context information of the user. In this scenario, Dimitrios' location within the office is important.
iii) *Display phone service context source:* In this case, a context source of the display phone service provides the location of the display phone and its availability.
iv) *D-Me context source:* The D-Me context source provides Dimitrios' location.

This work uses the Web Ontology Language (OWL) [15] to describe the context ontology. OWL is a semantic markup language that can be used to publish and share ontologies on the Internet. A resource in OWL is represented as a class, and the relationship between resources is shown using properties. To represent context information, a context source must create an individual of the class and connect two individuals by the property. Figure 4 shows how the context ontology has been mapped to the context information by a *Display phone service context source*. The *Display phone service* informs the CASD service about the context ontology it is referring to and a reference to the context source which provides the required context information in the form of a graph.

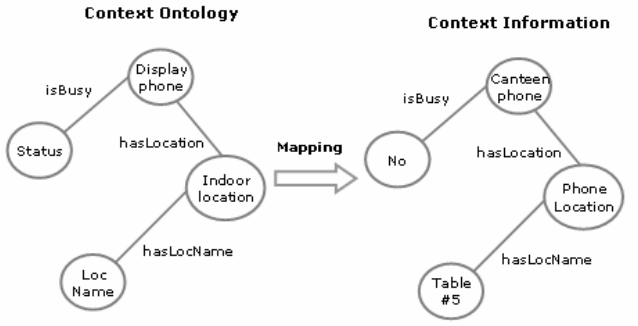

Figure 4. Mapping context ontology to the context information

Similar to the Display phone service context source, the *D-Me context source* maps context ontology to represent location information of Dimitrios. The *Context Updater* component of the D-Me context source updates location information that it obtains from a sensor.

The query sent by D-Me to the CASD service is expressed in SPARQL [18]. SPARQL is a flexible query language for OWL graphs. The query says: *'Provide me a display phone service the location of which is the same as Dimitrios' location and the service is not busy'.*

After receiving this query, the CASD service collects the context information of all the basic Display phone services (those received by the service directory) and stores it in a data structure called as *service graph*. The *Display phone service graph* is further sent to the reasoner for matching services which have same location as that of Dimitros and which are not busy. This work uses Jena [10] for matchmaking, which is a framework for building semantic web applications. It includes a rule-based inference engine, support for ontologies, a querying mechanism, and a persistent storage capability. When a client issues a persistent service discovery request, Jena's persistent storage capability is used. For this purpose, Jena stores the service context information in a MySQL database.

IV. CASD Implementation And Further Work

In this work, CASD has been realized using Jini [19] technology which follows the basic principles of SOA. The Jini infrastructure is built on top of the *Java* application environment. Jini uses Java *Remote Method Invocation* (RMI) as an underlying network protocol, e.g. RMI enables Jini clients to dynamically download code (*service proxy*) which is needed to access Jini services. The Jini infrastructure enables Jini services to register with *Lookup* services through *discovery* and *join* protocols. In this work, CASD service, other services in the network and context sources are Jini services. A service

directory is equivalent to the Jini lookup service. A reference to the context source is *serviceID*, which is obtained when a context source registers with Jini lookup service. The services provide information about their context sources and context ontology they are referring to, using *Service Entry* feature of Jini. In Jini, a *Service Entry* provides additional information about the service. A context source uses the *remote eventing* mechanism provided by Jini to notify changes in context information. A client (in this case the CASD service) interested in the context information implements a *remote event listener* interface to receive *remote events*. The CASD service also subscribes to the Jini lookup service so that it can be notified when new basic services are registered.

As shown in Figure 4, a mobile context source participates as a service in the fixed network using the Mobile Service Platform (MSP). The MSP design is based on the Jini Surrogate Architecture Specification [20], which enables devices that can not directly participate in a Jini Network to join a Jini Network (with the aid of a third party). MSP consists of an *HTTPInterconnect* protocol to meet the specifications of the Jini Surrogate Architecture and provides a custom set of APIs for building and running services on a mobile device. The context service in the fixed network exports a service proxy to the Jini lookup service. The CASD service uses this proxy to communicate with a context source, as shown in Figure 2.

V. Summary and Future Work

This paper presents our ongoing work in the area of augmenting *traditional* service discovery mechanisms with context-awareness. It introduces the concept of *persistent* service discovery. This mechanism promises to simplify the design of clients in pervasive environments as they need not actively search for the *best* services when their context changes. This added simplicity is due to the fact that they will be dynamically notified of better service matches as they become available. Further work needs to be done in determining the network and computational impact of this functionality on both the client and the server.

We plan to extend the existing CASD system for deployment as a bundle operating within an OSGi runtime. By moving the CASD into an OSGi home gateway, we benefit in three ways. Firstly, the CASD is able to be easily updated since OSGi provides simple mechanisms to manage the lifecycle of its deployed *bundles*. Secondly, having the CASD *hosted* in a home gateway also provides simple access to the home's network (be it fixed and/or wireless). Thirdly, the CASD can be re-engineered to easily make use of other bundles that can aid it in its task, such as regular service discovery and the *Context Comparator Service*. The *Context Comparator Service* will leverage some of the experience we have already gained with the use of Jena. It will make use of the context ontology and the client's query for service discovery to evaluate both the client and services contexts for aspects such as *closest, fastest, cheapest* etc.

Acknowledgements

The authors would like to thank Cristian Hesselman, Henk Eertink, Sorin Iacob and Aart van Halteren who contributed to the development of some of the concepts outlined in this paper. This work has been conducted as part of the Amigo project (IST-004182), which is partially funded by the European Commission.

References

[1] E. H. L. Aarts, et.al., "Ambient Intelligence", First European Symposium, EUSAI 2003, Veldhoven, The Netherlands, November 3.-4, Springer 2003.

[2] T. Broens et. al., "Context-Aware, ontology based, service discovery", Proceedings of the European Symposium on Ambient Intelligence 2004 (EUSAI'04), LNCS 3295, Eindhoven, the Netherlands, 2004.

[3] C. Campo et. al., "PDP and GSDL: A New Service Discovery Middleware to Support Spontaneous Interactions in Pervasive Systems.", PerCom Workshops 2005, pp. 178-182.

[4] L.Capra, S. Zachariadis and C. Mascolo. "Q-CAD: QoS and Context Aware Discovery Framework for Adaptive Mobile Systems". University College London, Tech Report RN/04/18. Sept. 2004.

[5] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research", Technical Report TR 2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[6] D. Dey, "Providing Architectural Support for Context-Aware applications", PhD thesis, Georgia Institute of Technology, 2000.

[7] C. Doulkeridis and N. Loutas, M. Vazirgiannis. "A System Architecture for Context-Aware Service Discovery". International Workshop on Context for Web Services (CWS'05), Paris, France, July 2005.

[8] K. Ducatel et. al., "Scenarios for Ambient Intelligence in 2010", ISTAG report, February 2001.

[9] A. T. van Halteren and P. Pawar, "Mobile Service Platform: A Middleware for Nomadic Mobile Service Provisioning", 2nd IEEE International Conference On Wireless and Mobile Computing, Networking and Communications WiMob 2006, Montreal ,Canada, June 2006.

[10] HP Labs, "Jena - A Semantic Web Framework for Java", http://jena.sourceforge.net/, October 2005.

[11] R. Jakkilinki, N. Sharda and I. Ahmad, "Ontology-Based Intelligent Tourism Information Systems: An overview of Development Methodology and Applications", Proceedings of Tourism Enterprise Strategies – 2005, Melbourne, Australia, July 2005.

[12] C. Lee and A. Helal, "Context Attributes: An Approach to Enable Context-awareness for Service Discovery," Third IEEE/IPSJ Symposium on Applications and the Intrnet, Orlando, Florida, January 2003.

[13] D. Nickull, "Service Oriented Architecture Whitepaper", Adobe Systems Inc., 2005.

[14] OSGI Alliance, "The OSGi Service Platform - Dynamic services for networked devices", http://www.osgi.org, 2006.

[15] OWL, "OWL Web Ontology Language Reference", http://www.w3.org/TR/owl-ref/.

[16] Z. Salvador et. al., "Architectures for ubiquitous environments", IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2005, Volume 4, pp. 90-97, Montreal, Canada, August 2005.

[17] B. Shishkov and P. Dockhorn Casta, "AWARENESS Service Infrastructure D2.10 - Architectural specification of the service infrastructure", https://doc.freeband.nl/dscgi/ds.py/Get/File-60592, Freeband Awareness project, 2005.

[18] "SPARQL Query Language for RDF", http://www.w3.org/TR/rdf-sparql-query/.

[19] Sun Microsystems, "The JINI Architecture Specification", http://www.sun.com/software/JINI/specs/ JINI1_2.pdf, December 2001.

[20] Sun Microsystems, "JINI Technology Surrogate Architecture Specification", http://surrogate.JINI.org/sa.pdf, October 2003.