*This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2007 proceedings.*

# Enabling Context-Aware Computing for the Nomadic Mobile User: A Service Oriented and Quality Driven Approach

Pravin Pawar[1], Aart van Halteren[2], Kamran Sheikh[3]
[1,3]Architecture and Services for Networked Applications Group
Department of Computer Science
University of Twente, The Netherlands.
{p.pawar, k.sheikh@utwente.nl}
[2]Philips Research, Eindhoven, The Netherlands.
aart.van.halteren@philips.com

*Abstract*— **Context-awareness (CA) enables the development of personalized applications for highly mobile and demanding users in the pervasive environments. These applications rely on the components responsible for context sensing, processing, storage and inference. Various context-aware computing infrastructures exist to shield the application developers from the complexity of developing these components. However, some additional concerns including the distribution of context sources in the pervasive environments, Quality of Context (QoC) and user privacy demand that a context-aware computing infrastructure should also handle these aspects.**

**This paper introduces our work in progress on the Context Distribution Framework (CDF) aimed at providing a service oriented infrastructure for the context-aware applications hosted on a mobile device. In the current version of CDF, we focus on three aspects: a) transparent off-loading of resource intensive context manipulation from the mobile device; b) selection of the suitable context sources based on the Quality of Context (QoC); and c) modeling mobile context sources as services.**

***Keywords-context-awareness, Quality of Context, context distribution, context sources, nomadic mobile context sources***

## I. INTRODUCTION AND MOTIVATION

Pervasive computing technology encompasses the users' surroundings by means of multiple independent sensors, actuators and computing nodes interconnected through wireless or wired connectivity. The users in a *pervasive environment* are often mobile; join and leave various networks and use multiple devices for communication. These users prefer to use applications tailored to their needs, location, time, user identity/profile and device capabilities [1]. *Context-awareness* enables the development of personalized applications. *Context-aware systems* adapt to the context of the user, application and their communication and computation environment, as well as to the changes to the context information over time.

In the context-aware system, an application relies on the components responsible for context sensing, processing, storage and inference [2]. There exists various middleware infrastructures (e.g. [2], [3], [4], [5], [6], [7], [8]) which provide support for the development of one or more of these components.

In this paper, we specially consider the case of providing context-aware computing support for the applications hosted on mobile devices such as a mobile phone or PDA. A *context source* provides necessary context information about the entity it is associated with [9]. The context sources are distributed in the pervasive environments. Among these, the mobile context sources display intermittent behavior because of the variable communication environment. To make the context information available for processing, storage and inference, a context-aware application (client) needs to perform resource-expensive tasks of locating these context sources, reasoning to select the context sources of interest and binding to these context sources. Provided that the mobile devices are poor in terms of the communication and computation resources as compared to their counterparts in the fixed network, the context-aware applications hosted on these devices cannot support these operations. Furthermore, there has been increasing concerns in the area of context-aware computing with respect to selecting the context source which provides context information with the desired *Quality of Context* (QoC) and handling user's privacy requirements at by the context sources. From the application developers view, these concerns need to be addressed for every context-aware application. To reduce the complexity of context-aware applications, improve maintainability and promote reuse, it is desirable that the components addressing these recurring design challenges should be provided by the *infrastructure* [5].

This paper introduces our work in progress on the *Context Distribution Framework* (CDF) aimed at providing service oriented infrastructure for the context-aware applications hosted on a mobile device. Section II of the paper elaborates the most important concerns for CDF. Section III presents current architecture of CDF and justifies our choice of service oriented architecture based design. Note that at this stage we do not address all the concerns listed in the Section II. Section IV discusses the technologies in use for CDF implementation. Section V provides a overview of the existing middleware infrastructures which support the development of context-

aware applications and shows the contribution of CDF in this area. Section VI summarizes the paper and provides pointers to the further work.

## II. CONCERNS FOR CDF

This section derives the most important concerns associated with the context sources and context information to be addressed by CDF.

### *A. Distributed and Intermittent Nature of Context Sources*

The context sources as well as client applications which make use of them are usually distributed physically as well as functionally. For example, in the health-care domain, an ad-hoc network consisting of a *Body Area Network* [10] and a mobile device provides the physical context (vital life signs) of the patient while a server operating in the fixed network predicts the probability of an epileptic seizure using the collected physical context. Another aspect is an intermittent behavior of the context source. E.g. when a mobile device experiences weak or no connectivity, a context source is likely to be not available. Moreover, the owner of the context source has ultimate control on it and may not choose to make it available continuously. CDF should be able to acquire the context spread in a variety of networks, distribute it to the applications and handle outage of the context sources.

### *B. Support for Quality of Context (QoC)*

Context information describing the real-world situation is inherently *vague,* e.g. it might not be known with 100% certainty or it might not describe the situation in enough details. Based on our experience with context-aware application development and current literature ([6][11][20]) we have identified the following QoC indicators:

1) *Precision* represents the granularity with which context information describes a real world situation. E.g. a doctor requires a patients' body temperature with at least three significant figures precision (such as 36.3°C).

2) *Freshness* denotes the time that elapses between the collection of context information and its delivery to a requester. E.g. a doctor requires that a patients' body temperature is not older than 1 hour.

3) *Temporal resolution* signifies the period of time to which a single instance of context information is applicable. This might vary due to the sampling rate of the context source. E.g. the temperature of a room collected every 8 hours is valid for a period of 8 hours after it is collected.

4) *Spatial resolution* symbolizes the precision with which the physical area, to which an instance of context information is applicable, is expressed. E.g. a building security system that keeps track of the number of people present in the building may provide this informtion with the spatial resolution of a room, a floor, a section of the building or the whole building.

5) *Probability of Correctness* corresponds to the probability that an instance of context information accurately represents the corresponding real world situation, *as assessed by the context source*.

It is preferable to model the QoC provided by context sources as a range (*minimum* and *maximum*). E.g. location with a precision of +/-10m. We adopt techniques for quantification of QoC indicators presented in [20]. Accurately determining values of QoC indicators requires empirical analysis and CDF delegates this responsibility to respective context sources. Determining values of QoC indicators for *primary* context sources, that collect information directly from physical sensors, is trivial as their capabilities and performance are mostly well documented. For *aggregated* context sources, where context from two or more sources is combined, computation of the QoC values for the resultant information needs to be part of the aggregation algorithm.

### *C. Representation of Context and Context Source Capabilities to Support Reasoning*

Context represents a particular knowledge which can be used to reason certain information. Depending on the scope of context-aware application and its users, context encompasses a variety of information. There exists a variety of research literature identifying the nature of context and providing its taxonomy [12]. E.g. the *physical context* such as location, light, movement, touch is measurable by hardware sensors, whereas the *logical context* such as user's goals, tasks, business processes and user's emotional state is captured by monitoring user interactions or specified by the user [12], or can be inferred from the underlying physical context [9]. A *multi-disciplinary* model [13] classifies context as *meaningful context* (related to the user's primary high-level goal e.g., to catch a train) and *incidental context* (unrelated to the user's primary high-level goal e.g., being caught in a sudden downpour). This signifies that the nature of context is *multifaceted* and has diverse representations.

Context representation uses the concepts from the field of *knowledge representation*. This allows representing context using symbolic structures that enable machine-based reasoning on context. Some possible choices include *key-value models*, *markup scheme models* (e.g. PIDF [14]), *UML models*, *logic based models* and *ontology based models* [12] among others. To help selecting an appropriate context source all the entities i.e. CDF, context sources and client should adhere to the common representation of the context and capabilities of the context source.

### *D. Users' Privacy Enforcement*

The QoC with which context is provided to the client also reflects its *privacy sensitiveness*. E.g. to protect user privacy, the time when an employee left the office may be shown only with *day* precision (e.g. *04-Sep-2006*). Therefore, *coarse-grained privacy policies* that evaluate to a *boolean* choice of whether the client may be given access to certain information or not are not enough. The CDF should expose *easy-to-use interfaces* so that users can specify privacy policies to obfuscate detailed context information provided to requesters (effectively reducing the QoC).

### *E. Knowledge of the Capabilities of the Context Source*

To select the context source(s) of interest for a client, the minimum necessary information is:

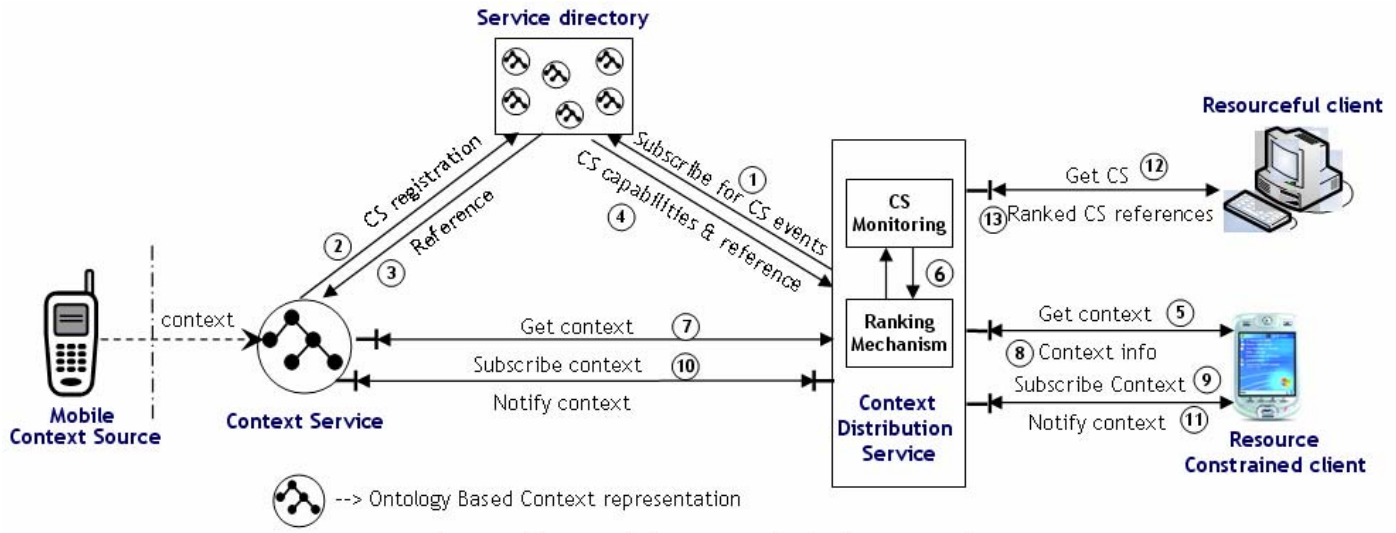


Fig. 1: Architecture of the Context Distribution Framework

a) The entity of which the context source provides information;

b) The type of context provided by the context source; and

c) Client's QoC requirements.

To select the context source which fulfils these requirements, it is necessary to map this information to the one provided by the context source. We consider these factors as the *capabilities* of the context source. It is likely that multiple context sources have similar capabilities (e.g. a context source located on the user's mobile device and another context source located in the operator network both provide the location of the same user). CDF should provide necessary mechanisms which allow publishing the capabilities of context sources as well as *matching* these capabilities against the client requirements.

### *F. Performance, Scalability and Fault Tolerance*

The operations required to match the capabilities of context sources against the client requirements are expected to incur some delay. However, it is desired that this overhead is within the acceptable limits. Furthermore, CDF should remain operational even when there is considerably large number of potentially appropriate context sources and a potentially large number of clients. We envision that CDF should achieve *organizational*, *geographical* and *numerical scalability* with sufficient performance.

## III. ARCHITECTURE OF CONTEXT DISTRIBUTION FRAMEWORK

Considering the distributed and intermittent nature of context sources and that the owner desires to have flexible control on the context sources, it is advantageous to model them as *services*. We consider a service to be a unit of *well-defined functional behavior* (in syntax and semantics) that is offered by a software entity for use by other software entities [15]. *Service Oriented Architecture* (SOA) paradigm allows flexible publishing and utilizing service offering and usage on the *Internet*. *Dynamic Service Binding* is a concept provided by SOA and adds to the popularity of SOA, because a service user is not required to be aware of the presence of a service a-priori [17]. The ability to *bind* to a service, as the SOA advocates, allows mobile devices to participate to relevant services on-demand. Modeling context sources as services provide the flexibility to a client to perform on-the-fly queries for a context source that best matches its requirements.

In SOA, a service provider *registers* the service description in the *service directory*. A client interested to access the service obtains the required information from the service directory using a process known as *service discovery* [16].

Considering SOA based design for CDF has certain advantages:

a) The owner of a context source has flexibility to publish them as desired;

b) A CDF client can *dynamically bind* to a context source that best matches its requirements;

c) A CDF context source as a service provides standardized functionality to its user;

d) It also handles the problem of dealing with intermittent context sources because the service directory removes the reference of the context source once it is unavailable; and

e) Another context-aware application components such as those responsible for storage of context information could also be developed as higher level services on top of context sources (please refer to *service composition* proposed in [15]).

Considering the advantages of SOA, we model the functionality offered by CDF to the clients also as a service referred to as *Context Distribution Service* (CDS). The context sources as well as CDS register with the service directory so that they can be discoverable. Figure 1 shows the architecture and components of CDF.

### *A. CDF Context Representation Model*

CDF includes support for *ontologies* to represent context information for various reasons (Refer Section II.C on

requirements). Compared to other techniques, ontology is the most expressive and widely adopted knowledge represented technique exploited in diverse areas. By using ontologies for context representation, *context reasoning* to derive high-level context from low-level context becomes possible [3]. There is a variety of open source software (e.g. *Jena* from HP Labs) available which provide functionality to *create* ontologies, *import*, *extend* and *merge* them as well as *reason* about the information represented using ontologies. The use of ontologies enables computational entities and services to have a common set of concepts and vocabularies for representing knowledge about a domain of interest, while being able to interact with each other. Ontologies are also beneficial for the re-usage of knowledge, as several ontologies from various sources can be integrated to describe the specific domain [18]. This helps in reusing context concepts specified in various domains. For information on how to develop ontologies to represent context refer to [18].

### *B. CDF Context Sources*

One of the popular ways to represent all the possible context types of an entity is by modeling it as a *hierarchical structure* in which the *nodes* specify (sub-) classes of information. CDF distinguishes between two types of context sources: viz. a *primary context source* (i.e. providing context specified by some node e.g. location of a person) and *aggregated context source* (i.e. providing context specified by multiple nodes e.g. location, agenda and heart-rate of a person). An aggregated context source may also collect same type of context from multiple context sources with *different QoC* and perform certain reasoning to *refine QoC* of the context it provides. E.g. an *aggregated context source* of a person can use three context sources providing location using *GPS*, *triangulation technique* and the one located in the *operator network*. Sometimes, though a context source interacts with multiple primary context sources, it may not be always an aggregated context source (but possibly *translation* and *interpretation* context source [4]). In such case we consider it as a primary context source.

In the current CDF architecture, CDS subscribes to the service directory to get notifications (*interaction 1 in Figure 1*) when a context source registers in the service directory (*interaction 2*). The *registration information* also includes the capabilities of the context source. A context source is identified in the service directory using a unique reference (*interaction 3*) which is sent to the CDS along with the context source capabilities *(interaction 4)*. CDS can later use this reference to get the information required to invoke context source from the service directory. Currently, the context source capabilities include:

a) The type(s) of the context source (primary/aggregated);

b) The entity of which a context source provides context information,

c) The context type(s),

d) Reference to the context ontology it refers to and

e) The QoC values of the context information (optional).

A context source provides the following methods:

1. *getContext()* allows a client to obtain context information.

2. *subscribeContext()* allows a client to subscribe for the context updates. The client should provide a callback interface over which the context change notification is sent.

An important aspect of CDF is that the context sources hosted by the mobile device (refer Section II.A) can also participate in a service discovery network using the concept of *Nomadic Mobile Service* [19]. We refer to such context source as *Nomadic Mobile Context Source*. A nomadic mobile service hosted by the mobile host participates in the service discovery network through its *proxy* in the fixed network [19]. Using a proxy improves the reliability, performance and responsiveness of a nomadic mobile service.

### *C. CDF Matchmaking and Ranking Mechanism*

After receiving the information from the service directory about a new context source, CDS analyzes its capabilities to create a data structure represented as: *context source (<reference>, <entity>, <context types>, <QoC>, <aggregated/primary>)* and stores it in a database locally. In our experience, unlike QoS metrics used in multimedia and telecom applications, the range of values of QoC indicators offered by a context source do not change very frequently. This is because while QoS metrics depend on highly dynamic factors like network traffic conditions, the QoC-based capability of a context source depends on the physical conditions (hardware performance, deployment etc) of the underlying sensors, which rarely change. Thus, there is no need for a mechanism that monitors the QoC provided by context sources.

On receiving a request from the client, CDS performs initial selection of context sources by matching the *entity* and *context type* information. If CDS finds any context sources it ranks them based on the similarity in *offered QoC* vs. *desired QoC* and the type of context source.

CDF offers support for the representation of QoC ranges (refer Section II.B) using the quantification techniques proposed in [20]. Considering that the ranges of the *offered QoC* and *desired QoC* are represented as a vector *QoC* ($<P_{min} - P_{max}>$, $<F_{min} - F_{max}>$, $<S_{min} - S_{max}>$, $<T_{min} - T_{max}>$, $<C_{min} - C_{max}>$) where *P, F, S, T* and *C* represent *precision*, *freshness*, *spatial resolution*, *temporal resolution* and *probability of correctness* respectively, the problem of finding the context source providing *nearest desired QoC* becomes the problem of determining distance between the QoC vectors. A *distance function* (e. g. *cross products, covariance and correlation*) takes a pair of vectors and returns a small value for matching vectors and a large value for distant vectors. CDF uses *Euclidean distance function* [21] to calculate distance because it is widely adopted. It is possible that the context sources do not specify their QoC. In such case, the context source which specifies QoC is always preferred. An aggregated context source is preferred over primary context source because it has a higher level view of QoC. A context source is ranked based on its *score* calculated using the above criteria.

### D. Context Distribution Service

*Context Distribution Service* (CDS) provides the following methods:

1) *getContext():* allows a client to obtain the context information from the most appropriate context source ranked according to *CDF ranking mechanism*. The client should specify the following (*interaction 5 in Figure 1*):

a) The context ontology it is referring to;

b) The desired context types;

c) The entity of which the context information is required;

d) Desired QoC for each context type (optional).

The CDS selects the most appropriate context source (*interaction 6*), obtains context from the context source (*interaction 7*) and provides it to the client (*interaction 8*). In case the selected context source is not available, then CDS selects the second best context source and so on.

2) *subscribeContext()*: allows a client to subscribe to the context information. CDS subscribes to the most appropriate context source (*interaction 10*) and relays context notification to the client (*interaction 11*) as it receives from the context source. When the selected context source becomes unavailable (as observed from the timeout or receiving notification from the service directory), CDS selects the second best context source and so on.

3) *getContextSources():* allows a client to obtain a list of context sources (*interaction 12 and 13*) ranked according to *CDF ranking mechanism*. It is the responsibility of the client to choose a context source and perform fault handling in case the selected context source is not available or becomes unavailable after some time. This method is of interest to the resource-rich clients who further want to append CDF ranking mechanism with their own mechanisms.

### E. Service Directory

We assume that a *service directory* provides the functionality for the registration of services and is aware of the existence of a service using certain mechanism (e.g. *leasing*, *heart bits*) to monitor the aliveness of a service. The *Context Source (CS) Monitoring* module of CDS subscribes to the service directory for the following:

a) Existence of the new context sources as they become available in the network;

b) To know the unavailability or the failure of a context source.

Whenever a context source become unavailable, the service directory informs to *CS Monitoring* module after which its reference is removed from the database.

## IV. Technical Realization

In the current implementation prototype of CDF, various services i.e. context sources and a CDS have been realized using Jini technology [22] which follows the basic principles of SOA. The *Jini lookup service* acts as a service directory. A context source is uniquely identified by a *serviceID*, which is obtained when a context source registers with the Jini lookup service. The context sources provide capabilities information using *Service Entry* feature of Jini. A context source and CDS uses the Jini *remote eventing* mechanism to notify changes in context information.

The *Mobile Service Platform* (MSP) proposed in [19] is a middleware that facilitates the development and deployment of nomadic mobile context sources. The MSP design is based on the *Jini Surrogate Architecture Specification*. Using MSP, a service hosted on a mobile device can participate as a *Jini service* in the Jini network by means of its *surrogate*. The *surrogate* functions as a proxy for the device service and is responsible for providing a service to the clients. Because of the tight coupling of MSP with Jini and to leverage the benefits of mobile context sources, Jini is also a choice for CDF implementation. The implementation of CDF uses the ontology developed as a part of Amigo project [18].

## V. Related Work

Providing standardized support for context distribution is an active research area as it deals with the vital task of making context information available to the context-aware applications. [6] discusses the issues involved in a context service. It discusses that the context service should handle aspects such as privacy, scalability, extensibility, synchrony and QoC. [2] argues that a context-aware middleware should address the challenges related to heterogeneity, mobility, scalability, privacy, traceability and control, fault tolerance and ease of deployment and configuration. The *PACE* middleware proposed in [2] consists of a context management system that provides aggregation and storage of context information, in addition to performing query evaluation. The *SOCAM* middleware [3] provides a set of services that perform context acquisition, context discovery, context interpretation and context dissemination. [3] also proposes to use a context model based on ontologies to share, understand and reason on the context information. The *MobiPADS* middleware [7] uses an eventing mechanism to notify change in the context information. The context events are used to support active service deployment and reconfiguration of the service composition in response to environments of varying contexts. A context-aware infrastructure proposed in [4] also uses context service to provide context information. It provides the necessary functionality including context acquisition, aggregation and notification mechanism. In [4], a context service can be elementary context service, aggregation service, translation service, interpretation service or abstract context service. The *Gaia* middleware [8] includes a context service which allows applications to query and register for particular context information. The context infrastructure consists of the components necessary for context acquisition and inference.

The differentiating aspects of CDF as compared to the related work described above are as follows:

a) CDF offloads the mobile client from the responsibility of locating, selecting, binding to and handling error conditions due to sudden disappearance of a context source,

b) CDF includes a reasoning support to rank context sources according to the QoC,

c) CDF provides support for modeling the context sources hosted on mobile devices as services and making them available in the service discovery network.

[21] also proposes to use QoC to select the right context source. In [21] a utility function is provided by the application and is evaluated in the middleware. However the differentiating aspects of CDF for making use of QoC are as follows: a) use of ontologies for QoC representation; b) well defined set of QoC parameters; and c) pre-defined utility function in the middleware.

## VI. Summary And Future Work

The Context Distribution Framework (CDF) provides infrastructural support for context-aware applications using SOA approach. CDF monitors the context sources in a variety of networks and provides service to the client to obtain the context information of interest. CDF further offloads a mobile client from the responsibility of selecting the context source of interest and includes a *fail-safe mechanism* to handle sudden disappearance of a context source. The CDF's *ranking mechanism* based on QoC is designed such that the client's QoC requirements are met. CDF also provides support for context sources hosted on a mobile device so that they can participate in the service discovery network.

In the future, CDF will offer multiple levels of service (e.g. *Gold, Silver* and *Bronze*) to its clients, representing different levels of QoC. These levels would get translated to specific values for each QoC indicator for each context type by the CDF. Thus, client applications are offered a relatively static set of choices that are translated to QoC values that are different for each context type. The current version of CDF has no support *user's privacy enforcement* (refer Section II.D). Furthermore, we have not done the performance evaluation of CDF (refer Section II.F). We are in the process of researching these concerns for CDF. Finally, scenario based validation of CDF will be done in the health-care domain.

## Acknowledgment

This work is supported by Freeband Awareness project (under grant BSIK5902390) and Amigo project (IST-004182, partially funded by the European Commission). The authors thank their colleagues in these projects who have contributed to the work described in this paper.

## References

[1] C. Doulkeridis, N. L., M. Vazirgiannis, *A System Architecture for Context-Aware Service Discovery*. International Workshop on Context for Web Services (CWS'05), Paris, France, 2005.

[2] Henricksen, K., J. Indulska, et al., *Middleware for Distributed Context-Aware Systems*. On the Move to Meaningful Internet Systems 2005, Agia Napa, Cyprus, 2005.

[3] Gu T., Punga H. K., Zhang D. Q., *A service-oriented middleware for building context-aware services*. Journal of Network and Computer Applications archive, Volume 28, Issue 1 January 2005.

[4] Zhang X., Liao J. and Liu J., *Open Middleware-Based Infrastructure for Context-Aware in Pervasive Computing*. Computational and Information Science: First International Symposium, CIS 2004, Shanghai, China, December 2004.

[5] Hong, J.I., Landay, J.A., *An Infrastructure Approach to Context-Aware Computing*. University of California at Berkeley, 2001.

[6] Ebling M. R., Guerney D. H. and Lei H., *Issues for Context Services for Pervasive Computing*. In Proceedings of the Workshop on Middleware for Mobile Computing. IFIP/ACM, 2001.

[7] Chan A. T. S., Chuang S. N., *MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing*. IEEE Transactions on Software Engineering, vol. 29, no. 12, December 2003.

[8] Roman, M. et al., *Gaia: A Middleware Infrastructure to Enable Active Spaces*. in IEEE Pervasive Computing, pp. 74-83, October 2002.

[9] 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.

[10] Konstantas, D., Bults, R., Wac, K., Halteren, A. V., *Mobihealth D2.6 - Final, Exploitation Ready MobiHealth BAN*. MobiHealth project (http://www.mobihealth.org), April 2004.

[11] Buchholz T., Küpper A. and Schiffers M., *Quality of Context: What it is and why we need it.* Proceedings of the Workshop of the HP OpenView University Association, Geneva, 2003.

[12] Baldauf, M., Dustdar, S., Rosenberg, F., *A Survey on Context-Aware Systems*. International Journal of AdHoc and Ubiquitous Computing, 2006.

[13] Bradley, N. A., & Dunlop, M. D., *Towards a multidisciplinary user-centric design framework for context-aware applications*. Human-Computer Interaction, Vol. 20, No. 4: pages 403-446, 2005.

[14] Sugano H., Fujimoto S. et. al, *Presence Information Data Format (PIDF)*. IETF Network Working Group, Internet Draft, 2003.

[15] Cristian Hesselman, Andrew Tokmakoff, Pravin Pawar, Sorin Iacob, *Discovery and Composition of Services for Context-Aware Systems*. 1st IEEE European Conference on Smart Sensing and Context, Enschede, The Netherlands, October 2006.

[16] Pravin Pawar, Andrew Tokmakoff, *Ontology-Based Context-Aware Service Discovery for Pervasive Environments*. 1st IEEE International Workshop on Services Integration in Pervasive Environments (SIPE 2006), Co-located with IEEE ICPS 2006, Lyon, France, June 2006.

[17] Dokovski N., Widya I., van Halteren A. T., *Paradigm: Service Oriented Computing*. Freeband/AWARENESS/D2.7b, http://awareness.freeband .nl, December 2004.

[18] Kalaoja J., Kantorovitch J. et. al., *Detailed Design of the Amigo Middleware Core Service Modelling for Composability*. http://www.hitech-projects.com/euprojects/amigo/deliverables/ Amigo_D3_1a_v1.0.pdf, September 2005.

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

[20] Sheikh K., Wegdam M., van Sinderen M., *Middleware Support for Quality of Context in Pervasive Context-Aware Systems*. Middleware Support for Pervasive Computing Workshop at the 5th Conference on Pervasive Computing and Communications. New York, USA, 2007.

[21] Huebscher, M. and J. McCann. *An adaptive middleware framework for context-aware applications*. Personal and Ubiquitous Computing archive 10(1): 12 – 20, 2005.

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

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