Towards ad hoc contextual services for pervasive computing

Damien Fournier, Sonia Ben Mokhtar, Nikolaos Georgantas, Valérie Issarny

Towards Ad hoc Contextual Services for Pervasive Computing

Damien Fournier, Sonia Ben Mokhtar, Nikolaos Georgantas,Valérie Issarny
INRIA
UR Rocquencourt
78153 Le Chesnay Cedex, France
{Damien.Fournier, Sonia.Ben_Mokhtar, Nikolaos.Georgantas, Valerie.Issarny}@inria.fr

ABSTRACT

Context-awareness is a key challenge for pervasive computing, as it is a prime requirement towards delivering applications to users in a way that best matches user requirements, digital resources availability and physical conditions. However, enabling anytime, anywhere context-awareness, as targeted by pervasive computing, is further challenged by the openness of the environment, which requires making available context information in various computing environments. This then calls for the ad hoc networking of context sources and of context-aware applications, so that applications may always benefit from a context knowledge base, although it may be more or less rich, depending on the specific environment. Building upon the context management literature, and the Service-Oriented Architecture (SOA) paradigm that is a major enabler of open ad hoc networking, this paper sketches key context-aware system concepts that need be incorporated in the SOA style towards enabling context-aware services for pervasive computing.

Categories and Subject Descriptors

H.1 [**Information Systems**]: Models and Principles; D.2.11 [**Software Engineering**]: Software Architectures

General Terms

Interoperability

Keywords

Context-awareness, Pervasive Computing, SOA.

1. INTRODUCTION

Context awareness and management are challenging requirements for pervasive computing decisively affecting the user's experience [19]. Nowadays, many applications using context information are proposed (e.g., healthcare monitoring, personalized service delivery). Context management gives to applications the aptitude to be aware of user characteristics, system behavior and state of the physical environment [8]. These applications, qualified as context-aware, are able to adapt dynamically their behavior according to user requirements and/or environment.

Emergence of service-oriented computing as a well-serving paradigm for pervasive environments has indicated a tight association between pervasive context and services. To this end, several service-oriented middleware architectures have been proposed dealing with one or more of: context identification and description; context retrieval, processing and storing; reasoning on context; context-aware service description, discovery and selection; and context-aware service adaptation (e.g., see [2] for a survey). In these architectures, a number of functional entities have been identified participating in the generation, management and use of context. Context sources (e.g., sensors) provide a specific type of context information. Context aggregators and interpreters compile and process context information to make it available in an elaborated, added-value form. Context repositories organize and make context information accessible, further supporting reasoning for answering complex context-related queries. Context-aware services and their clients are matched and — if needed — adapted to each other by context-aware service discovery and adaptation mechanisms.

A key point to all aforementioned functionalities is the description of context, where the paradigm of choice in most approaches is ontologies coming from the knowledge representation domain. Ontology languages are employed to enable common understanding of context semantics and reasoning on complex relations between contextual concepts. Further, of major interest is the distribution of context-aware architectures. We observe two directions for managing context information. First, context management support may be limited to a specific physical environment, such as home or office. We define these physically delimited environments as pervasive spaces. In this case, context information is conveniently managed by a (logically) centralized system, while mobile devices (PDAs, smart phones) supporting user applications are consumers of such information. The second approach lies in delegating context management to (end-user) devices so that devices have access to context information in any environment. In this case, each pervasive device is responsible of its context. Each device then supports local context management, independent of any infrastructure, possibly collaborating in an ad hoc way with peer devices to enrich its context knowledge. Accordingly, context-aware service discovery, selection and

MW4SOC '06, November 27-December 1, 2006 Melbourne, Australia

adaptation may be executed in a centralized or decentralized way, where the former asks for a context management infrastructure provided by the pervasive space, and the latter relies on the individual pervasive devices. Architectural and functional distribution further affects consumption of context information. RPC or event-based interaction may be employed for retrieving specific context information and for invoking context-aware services, while ontology-oriented query languages may be employed for conveying complex context-related requests (e.g., see [17]).

Understanding context-aware architectures for pervasive computing with respect to all the above discussed features is the main focus of this paper. First, we briefly survey related research approaches, summarizing key aspects underlying context awareness, from context sources to context management (Section 2). A major outcome of our survey is that all reviewed approaches commonly assume – each – a homogeneous context-aware architecture, where immediate compatibility of functional and knowledge-related entities is assured by design. However, the openness and high dynamics of pervasive environments make such an assumption too restrictive. In these environments, heterogeneity may arise in any of the above discussed features of context-aware architectures. Hence, we envision a service-oriented middleware approach allowing for ad hoc context-awareness in pervasive environments, where context-related entities – all taking the form of contextual services – dynamically self-organize and self-adapt to optimally exploit available, possibly heterogeneous, contextual resources at the specific time and place. Major challenges in this approach are adequate semantic description of contextual services and principally of their role in context awareness (i.e., from context source to context consumer), and, based on this description, self-organization including contextual service discovery, composition, adaptation and interaction. As a first step towards this objective, we elaborate a conceptual model in UML representing key context-aware system concepts as elicited from our survey; we further inject in this model our vision of ad hoc context-awareness (Section 3). We finally conclude in Section 4 presenting our plan of ongoing and future work.

2. CONTEXT AWARENESS

A context-aware pervasive environment is made of software applications that execute opportunistically according to user presence and networked resources. Applications are further able to adapt to the physical space and to available hardware and software resources, always aiming at providing best user experience. Devices of the context-aware pervasive environment then exchange diverse context information through the network.

2.1 Taxonomy of Context Information

Context information is usually classified in three subsets, called *context domains* [20]: user domain, system domain and physical domain. The *user domain* provides knowledge enabling applications to adapt their behavior according to the profile of their users. For example, if the user is blind, applications can sense this information by getting the description of the user handicap and then adapt their user interface so that speech is chosen as the interaction modality. As another example, a context-aware video player can read the user profile to check if the user is allowed to watch the video content according to the user's age. User-related context information further subdivides into *subjective* and *objective user context*. The former defines the user's *personality* and *psychology*, corresponding to user mood and feeling (fear, anger). The latter includes the following information:

- *Personal information* provides data allowing identification of, and communication with, the user such as, first name, last name, birth date, home address, etc. Communication-related information accounts for the various communication networks (providing, e.g., phone number, email address, instant messaging nickname), further enabling to choose the most appropriate communication means according to network connectivity and other relevant context data.
- *Physiology and condition information* defines height, weight, eyes color, health (ill, tired, stress), etc., which may in particular be exploited by healthcare or diet coach applications.
- Finally, *agenda-related information* defines user activities over time, enabling the system to adapt communication/interaction modalities (e.g., switching off ring tone while attending a meeting), and in general inferring various information and actions (e.g., knowing about the user's location).

The *system domain* describes digital, software and hardware resources available to users. Applications use description of devices features and resources characteristics to adapt system behavior, alter content, or modify user interfaces [4]. For example, a video player application can display content on the largest screen available in the room where the user is located but once another user enters the room, the content is displayed on the user's mobile device and adapted according to screen resolution. In order, to adapt application behavior according to devices, the system domain usually combines description of hardware and software components available in the context-aware environment. Typically, this description specifies:

- *Processing power* (e.g., CPU type and frequency, RAM available, energy source and autonomy).
- *Hardware components*, including input/output peripherals attached to the device. Typically, their description provides information about screen quality, printer and scanner resolution, etc.
- *Network interfaces* to connect the device to the various networks, both wired and wireless. This description includes network interface name, network type, available protocols, hardware and logic address, and bandwidth.
- *Storage system* (e.g., partitions, files system, capacity).
- *Software components* like operating system and virtual machines.

Finally, the *environmental domain* deals with the description of location and of conditions of the physical environment. User mobility or natural variation of physical conditions (e.g., brightness or temperature) can impact application behavior within a context-aware environment. The environmental domain describes locations where users and devices are physically situated. It allows retrieving the user and possibly adapting the physical environment (with physical actuators) according to the user's preferences. The environmental domain subdivides into three sets of context information:

- *Physical geography* provides description of locations with room name, building type, postal address, absolute coordinate, for indoor environment, and GPS coordinate, road, for outdoor environment.
- *Physical conditions* provide description of outdoor weather condition (wind, sun, rain, temperature), and of sensing conditions such as brightness, ambient noise, humidity.
- *Chronology defines timing information such as time zone and time of the day.*

2.2 Context Representation

To be effectively managed, context information need be modeled in a uniform way so that it can be unambiguously interpreted and further combined. Six models dominate in the literature for context representation [21]:

- *Key-Value models* represent context information using a set of attributes and their associated values. It is the most simple data structure used in context-aware applications.
- *Markup models* enable structuring context information into a hierarchy. Tags describe context attributes and associated values. Using tags recursively permit classifying context into data subsets.
- *Graphical models* are suitable for expressing relationships between context entities, as illustrated in [11].
- *Object models* take benefits of the object-oriented paradigm to represent context information as object variables, and structure this information into object classes [12]. Object oriented models add the possibility to mix context processing with context data.
- *Logic models* represent context in terms of facts and rules. They are used to infer new statements about context information.
- *Ontologies* benefit of both logic and object orientation. They enable representing context using classes and explicit relationships among them.

Basically, to structure sensed values into context information, each piece of context data must be precisely characterized. Context values are usually described with metadata [13]. Main attributes associated with a context value are: (i) the context domain (see §2.1), and (ii) the context type that refers to subsets belonging to the context domain. Sometimes, the context type is written with a path expression, to describe the "type" attribute within a hierarchy [13, 7]. In order to add precision to context information, more sensing details are needed such as:

- The *context source* attribute refers to the sensing service that gives context values related to, e.g., physical sensor, device or application.
- Related to the context source, the *quality* attribute gives details about sensing mechanisms, usually related to accuracy, timeliness or confidence of context source. It can be used to adapt context processing and exclude inaccurate sources.
- The *metric* attribute defines the data type and unit of context values, enabling applications to instantiate and process context information.
- The *timestamp* attribute is essential to capture history of context values and infer information over a time period, or to check up-to-datedness.
- The *relationship* attribute enables linking correlated context information, making explicit impacts of sensed values on other context information, or defining interaction between users, devices and the physical environment. For example, when a user turns on his mobile phone, interaction of the user with the mobile device can be specified by the relationship "use".

Associated with values, metadata gives details about the nature of context information. Metadata is used by the context management system to process and deliver context information to applications. The context manager hides details of sensing to applications and provides valued context attributes according to application requests.

2.3 Context Management

Initial research on context-aware systems focused on building applications for a specific scenario or a specific context, leading to develop application-specific context manager. This approach is often illustrated by location-aware systems. For example, the Active Badge system forwards phone calls according to the user location in the Olivetti research laboratory [22]. Some of these applications are also implemented as tourist guide, such as Cyberguide [1] and the GUIDE project [6]. Due to their specific nature, these solutions provide limited support to generic context management that can be reused across context-aware applications.

To ease the development of context-aware applications, dedicated frameworks have been proposed. Those frameworks provide reusable context components that are responsible of data acquisition, aggregation and interpretation. Context components provide a public interface for accessing context data, while they hide details of context sensing. For example, the Context toolkit introduces context components, called context widget [8]. The contextor infrastructure further assists the composition of context components [18]. Specifically, the context information is managed by contextor components, which provide communication channels for the exchange and control of context data. The assembly of contextors is ruled by a data-flow model. The composition of contextors may be static (specified at design-time), semi-static (specified at run-time), or transient. In general, framework-based approaches for context management greatly ease development of context-aware applications. They abstract details of sensing and transform observables into values that are processable by applications. Still, making applications context-aware requires developing applications with the specific frameworks and further having the specific context components actually deployed in the environment.

Availability of context information may greatly be improved through the deployment of context servers within the network(s). Context servers may be requested for context information by any networked client application. Typically, this solution is based on an infrastructure for sensing and reporting context to the context server. The context server then stores and interprets context information reported by sensing applications deployed into a physical environment (such as home, office, hospital). This approach eases sharing of context information among applications, as illustrated by

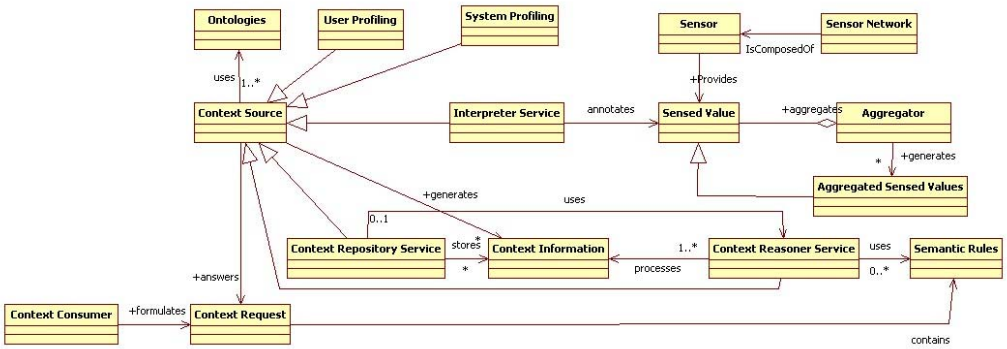

Figure 1: Context Sources

SOCAM [10] and Cobra [5]. The COntext Broker Architecture (CoBra) is composed of a context server and modules responsible of data acquisition [5]. Similarly, the Service-Oriented Context-Aware Middleware (SOCAM) introduces an infrastructure to manage context; it further undertakes a service-oriented approach with context acquisition and interpretation being deployed on an OSGI-based residential gateway [10].

We notice that more recent solutions extend the server-oriented approach to context management to a P2P architecture. Every computing device (PDA as well as desktop) becomes responsible of managing part of context information and peering with others to enrich the context knowledge base. This approach is illustrated by the AWARENESS project[1], where mobile and desktop devices are able to acquire, interpret and exchange context data. Compared to the base server-oriented approach, this empowers mobile terminals with the ability to support context-aware applications even when no context server is available in the networked environment.

3. CONTEXTUAL SERVICES

As surveyed above, context-awareness requires dealing with diverse sources of context information and making available related knowledge to applications. Then, developing context-aware applications can be tractable only if provided with adequate middleware support for managing context information, from acquisition to storage. Also, key middleware functions shall be made context-aware (e.g., context-aware service discovery [17, 9] as well as context-aware interaction). However, context-awareness shall not be bound to a specific software infrastructure or physical environment. Context-awareness shall be made available in most environments, thanks to the ad hoc networking of context sources and consumers. Such open networking is enabled by the service-oriented architecture paradigm, further backed by semantic Web technologies for making services interoperable. In addition, semantic services may be conveniently exploited in the pervasive computing environment through adequate lightweight middleware for wireless, resource constrained devices [16]. Building upon the above effort, we are investigating the development of *contextual services* for pervasive computing. Specifically, we call contextual service, a semantic service that realizes a context source and/or a context consumer. A contextual service further opportunistically networks/composes with other contextual services to offer a richer context knowledge base, as network connectivity allows. Towards the development of a contextual service-oriented architecture, and in particular related middleware support, this section elicits key architectural elements underpinning context-awareness, i.e., context sources (§3.1), context consumers and related communication protocols (§3.2), and context-aware services (§3). Elements are further introduced using UML models so as to highlight the relationships among them.

[1]AWARENESS Consortium. Context AWARE mobile NEtworks and ServiceS, `http://awareness.freeband.nl/`

3.1 Context Sources

Figure 1 models major context sources, as identified in Section 2.1. Specifically, context sources may provide information relevant to the user, system and physical domains.

The physical domain is dependent upon physical sensing of the environment by sensors, which may possibly be organized into dedicated networks. Sensed values related to the same physical data (e.g., temperature in a building sensed in various rooms) may further be aggregated (e.g., sum, average, ...). Given some sensed value, the *interpreter service* encodes it as a structured context data according to a target/reference ontology (see §2.2).

The *context repository* is the service responsible for the storage of context information. It typically stores information from the various context sources networked in the environment. Additionally, the context repository possibly manages a rich knowledge base by inferring new context information from the known context, thanks to the context reasoner. Typically, a *context reasoner* uses logic inference and reasoning mechanisms to deduce new context information. In general, services implementing context sources shall network and cooperate opportunistically so as to make the knowledge base available to applications as rich as possible. This may be coordinated by context repositories — if any — for the sake of resource savings. However, the actual cooperation protocol among context sources shall be determined dynamically according to various non-functional attributes, like availability, performance and privacy. All the context sources, i.e., user profiling, system profiling, interpreter, context repository, and context reasoner services, handle *context requests*. Such requests may simply provide context types as input. Alternatively, they may express complex

queries over the knowledge base using, e.g., SPARQL[2].

3.2 Context Consumption

As modeled in Figure 2, context sources may be accessed by context consumers using various protocols:

- *RPC-based interaction* is the most basic type of interaction, with the consumer requesting for a given context information. As such, it favors reactive context-awareness. In this case, the dynamics of the context information need be specified, using, e.g., timestamp attributes (§2.2).
- *Event-based interaction* allows for both reactive and proactive context-awareness, with the consumer being notified of requested context information upon given conditions (e.g., at regular time intervals, update since last notification).
- *P2P interaction* allows for advanced distributed context management, favoring ad hoc composition of context sources.

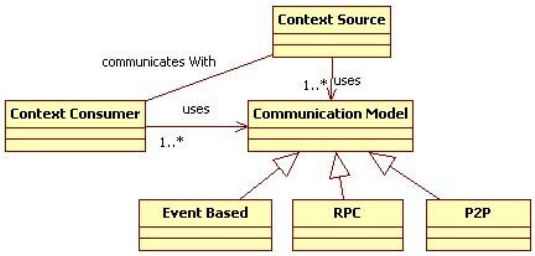

Figure 2: Context Consumption

3.3 Context Aware services

By accessing context sources, services of both the application and middleware layers may be made context-aware. This is modeled in Figure 3, which highlights context-awareness of the three base architectural components of SOA, i.e., service client and provider, and Service Discovery Protocol (SDP). Note that although we abstract each component as a single entity, they may be distributed as is in particular the case of the SDP.

We distinguish two complementary approaches to *context-aware service discovery*: *explicit* and *implicit*. In the former case, the request for the service specifies contextual requirements for the service provision, hence leading to select services according to applications requirements, as specified by the service client and/or provider [3, 17, 9]. Typical example of explicit context-aware service discovery is the case where users request to print their documents on the closest printer. In the implicit case, the context-aware SDP selects services according to functional and non-functional properties specified by the client, and further tunes the selection according to context knowledge. In both cases, the SDP needs to have access to context sources relevant to the client and/or service provider, which is to be enabled by the middleware. Then, this knowledge may be obtained reactively, proactively or

[2]`http://www.w3.org/TR/rdf-sparql-query/sparql-defns.html`

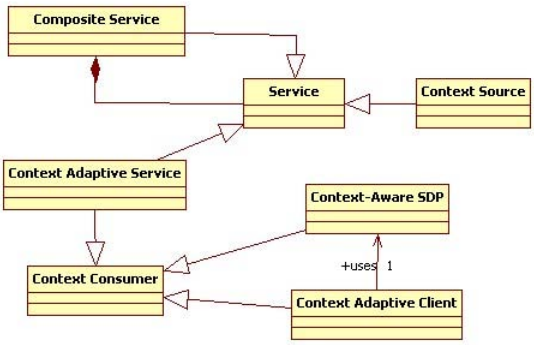

Figure 3: Context-aware Services

even via a cache, depending on its dynamics and the networked environment.

Context-adaptive service adapt their behavior according to context. We then distinguish 3 types of context-adaptive services: (i) the *context-specific service* can only be correctly provisioned in the specified context, (ii) the *context-dependent service* may be provisioned in various contexts but is bound to a specific context during a session, i.e., the service adapts its behavior at binding time, and (iii) the *context-aware service* continuously adapts its behavior according to context evolution. The third type of service relates to the development of dynamically adaptive software [23]. In all cases, the service interface must be enriched with the specification of the context in which the service can be correctly provisioned, which has to be matched against by the SDP through access to relevant context sources. Furthermore, the context may be monitored during the service session by the middleware, for triggering adaptation — if supported— or enforcing robustness of the system by detecting that the service can no longer be provisioned as expected. A context-adaptive service may be *composite*, in which case it is able to compose services according to the context of composition in addition to the functional and non-functional properties of composed services [14]. For example, if the user requests for a travel service, he can add constraint on the global composition in order to select the cheapest travel, or add constraint on composite flight services in order to select only best ranked companies worldwide.

Note that being a service, a context source may itself be developed as a context-adaptive service, possibly composite. For instance, a context repository uses a context-aware SDP for maintaining its knowledge base according to the context sources that join and leave the network(s) in reach. Finally, a *context-adaptive client* is dual to the context-adaptive service; it makes explicit the context in which the requested service is to be provisioned.

4. CONCLUSIONS

Service-oriented computing has emerged as a promising paradigm for pervasive computing. In particular, semantic services together with supporting lightweight middleware enable abstracting and composing on the fly networked resources, either wireless or wired, resource-constrained or resource-rich, mobile or stationary [16]. As such, semantic services enable a true interoperable, open network of ser-

vices that leave and join according to their mobility pattern. However, several challenges remain towards fully pervasive service-oriented computing. One such challenge is making services contextual, so that the pervasive computing environment gets knowledge about the context and the services adapt their behavior accordingly. In this direction, this paper has surveyed key architectural elements and in particular middleware-related ones that need be deployed in the environment for context-aware pervasive service-oriented computing. These take the form of contextual services that may be composed in an ad hoc way through opportunistic networking and combination of functionalities. We are now refining the proposed SOA style so as to develop a service-oriented context-aware middleware, which will enrich our current interoperable middleware for pervasive services [15].

5. ACKNOWLEDGMENTS

This work is supported by the IST PLASTIC Project — `http://www.ist-plastic.org`

6. REFERENCES

[1] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: a mobile context-aware tour guide. *Wirel. Netw.*, 3(5):421–433, 1997.

[2] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems.

[3] T. Broens. Context-aware, ontology-based, semantic service discovery, masters thesis, university of twente, enschede, the netherlands, 2004.

[4] T. Chaari, F. Laforest, and A. Celentano. Service-oriented context-aware application design. In *First International Workshop on Managing Context Information in Mobile and Pervasive Environments*, Ayia Napa, CYPRUS, 2005.

[5] H. Chen, T. Finin, and A. Joshi. Semantic Web in in the Context Broker Architecture. In *Proceedings of the Second Annual IEEE International Conference on Pervasive Computer and Communications*. IEEE Computer Society, March 2004.

[6] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of developing and deploying a context-aware tourist guide: the guide project. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 20–31, New York, NY, USA, 2000. ACM Press.

[7] P.-C. David and T. Ledoux. Wildcat: a generic framework for context-aware applications. In *Proceeding of MPAC'05, the 3rd International Workshop on Middleware for Pervasive and Ad-Hoc Computing*, Grenoble, France, Nov. 2005.

[8] A. K. Dey. *Providing architectural support for building context-aware applications*. PhD thesis, 2000. Director-Gregory D. Abowd.

[9] A.-R. El-Sayed and J. P. Black. Semantic-based context-aware service discovery in pervasive-computing environments. In *Proc. of the 1st Workshop on Services Integration in Pervasive Environments*, 2006.

[10] T. Gu, H. K. Pung, and D. Q. Zhang. A service-oriented middleware for building context-aware services. *J. Netw. Comput. Appl.*, 28(1):1–18, 2005.

[11] K. Henricksen and J. Indulska. A software engineering framework for context-aware pervasive computing. In *PerCom*, pages 77–86. IEEE Computer Society, 2004.

[12] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, and W. Retschitzegger. Context-awareness on mobile devices - the hydrogen approach. In *HICSS*, page 292, 2003.

[13] P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen, and E.-J. Malm. Managing context information in mobile devices. *IEEE Pervasive Computing*, 02(3):42–51, 2003.

[14] S. B. Mokhtar, D. Fournier, N. Georgantas, and V. Issarny. Context-aware service composition in pervasive computing environments. In N. Guelfi and A. Savidis, editors, *RISE*, volume 3943 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 2005.

[15] S. B. Mokhtar, N. Georgantas, and V. Issarny. Cocoa: Conversation-based service composition in pervasive computing environments. In *Proceedings of the IEEE International Conference on Pervasive Services*, June 2006.

[16] S. B. Mokhtar, A. Kaul, N. Georgantas, and V. Issarny. Efficient semantic service discovery in pervasive computing environments environments. In *Proceedings of Middleware*, December 2006.

[17] P. Pawar and A. Tokmakoff. Ontology-based context-aware service discovery for pervasive environments. In *Proc. of the 1st Workshop on Services Integration in Pervasive Environments*, 2006.

[18] G. Rey, J. Coutaz, and J. L. Crowley. The contextor: a computational model for contextural information. In *Wokshop Building Bridges Interdisciplinary Context-Sensitive Computing*, 2002.

[19] D. Saha and A. Mukherjee. Pervasive computing: A paradigm for the 21st century. *Computer*, 36(3):25–31, 2003.

[20] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.

[21] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, September.

[22] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, 1992.

[23] J. Zhang and B. H. C. Cheng. Model-based development of dynamically adaptive software. In L. J. Osterweil, H. D. Rombach, and M. L. Soffa, editors, *ICSE*, pages 371–380. ACM, 2006.